Resolving conflicts in agent-supported collaborative product development

Sreeram, R. T.

Award date:
2000

Awarding institution:
University of Bath

RESOLVING CONFLICTS IN AGENT-SUPPORTED COLLABORATIVE PRODUCT DEVELOPMENT

Submitted by R. T. Sreeram
for the degree of
Doctor of Philosophy
of the University of Bath
2000

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

Abstract

In a collaborative product development process, specialists from several life-cycle prespectives such as design and manufacturing need to interact. To support such collaboration, techniques based on software agents have emerged. These agents collaborate and carry out certain domain specific activities within these prespectives.

Agents in a collaborative process have individual preferences on certain aspects of the product. Conflicts occur when these agents differ in their preferences. It is important to resolve these conflicts through some rational means of negotiation. Most of the existing research in software agents has focused on software issues such as interoperability and communication rather than conflict resolution. In order to realise the potential of software agents in collaborative product development, there is need for domain-independent schemes to support rationalisation of agent's preferences and negotiation.

This research focuses on two inter-related threads of collaborative product development:

- Synthesis of software agents to support collaboration
- Conflict resolution between agents having varying preferences

The concept of Single Function Agents (SiFAs) has been adopted and modified using the Habermas's theory of action for synthesising agents with varying degree of complexity. An agent synthesis model has been proposed that uses SiFAs to construct complex Multi-Agents. For the resolution of conflicts between agents, a novel scheme based on game theory and dependency reasoning is developed. The new scheme first identifies the feasible design space and then finds single-point solutions within the space such that they represent a rational compromise between agents' preferences. The scheme is implemented by means of a negotiation agent and a reasoning agent.

The above scheme has been validated via two engineering case studies in an agent supported collaborative product development process:

- **Comparitive case study:** Collaborative design of a poppet relief valve to benchmark the proposed conflict resolution scheme in comparison with the previous research in negotiation.
- **Validation case study:** Collaborative design and manufacture of a Geneva mechanism to demonstrate the feasibility of the approach for agent synthesis and validate the novel scheme for conflict resolution involving both software agents and humans in a laboratory-based product development process.

The two case studies were mutually supportive in terms of examining the feasibility of software agent synthesis approach and validating the novel scheme for conflict resolution between agents. The conflict resolution scheme ensured rationality in decision making while preserving the preferences of agents. The key contribution of this research, extends the previous research with respect to conflict resolution in agent-supported product development while exploiting a new simplified agent synthesis model. The limitations of the present research and avenues for further research have been identified with respect to the scalability and generalisation to n-agent conflict resolution.

I wish to dedicate this thesis to:

My late grandparents

Smt. Lakshmi and Shri. Viswanatha Iyer, and Smt. Rukmini and Shri. Easwara Iyer;

My loving parents

Smt. Shantha Ramani and Shri. Ramani Iyer;

My loving wife

Shanthi.

Acknowledgements

I would like to express my sincere thanks to my research supervisor Dr. P. K. Chawdhry, for his patience, continued support and assistance throughout this research.

I am highly indebted to Professor A. J. Medland for his comments and constructive criticisms of my thesis. My thanks to Dr. E. S. Masuku, Mr. R. K. Pepler and Mr. A. J. Green for their assistance during my case studies. My sincere thanks to Dr. J. R. J. Rao (University of Houston, USA) and Dr. J. Wang (Feng-Chia University, Taiwan) for their helpful suggestions in specific aspects of negotiation. I owe a great deal to my friend Mr. R. F. Pannett, for his invaluable help in the context of proof-reading this thesis.

I would like to acknowledge the postgraduate studentship award by the University of Bath and the ORS award from the Council of Vice Chancellors and Principals (CVCP), London, for this research. I would like to take this opportunity to thank Ms. Heather Kellaway, Bath University Computing Services (BUCS), for offering me a part-time employment with BUCS which was very helpful in the finishing stages of this thesis.

I would like to express my warmest thanks to my parents **Mrs. Shantha Ramani** and **Mr. Ramani Iyer**, who ensured a sound education and up-bringing. Their continued support and encouragement has been instrumental in all my achievements thus far.

I would like to express my heartfelt thanks to my wife **Shanthi**, for her love and support throughout this research.

Finally, I would like to thank my sister, my brother, my brother-in-law, and my in-laws, for their constant support and understanding.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Since the beginning of industrial revolution, product development has evolved through several stages: from centuries old single-piece craft production to the present day Internet based collaborative product development. In order to fully appreciate the drivers for such a transition, it is necessary to review the origins of product development.

Craft production dates back to the 18^{th} century where production facilities were under-developed [Hanson (1984), Meier, *et al.,* (1994)]. Each product was manufactured with personal care using skilled labour that resulted in limited production and long lead time. In the first quarter of the 20^{th} century, the craft-based approach needed modification to suit a production process capable of meeting the increasing consumer demand and was referred to as mass production [Womack, *et al.,* (1991)]. Mass production was found to be capable of faster production rates, and ensured uniformity in both the aesthetics and quality of products. Meier, *et al.,*(1994) noted that fundamental changes which took place in mass production

also led to lean production, a concept introduced by Eiji Toyoda and Taichi Ohno in Japan during the early 1960's.

Lean production was viewed as a major improvement with reference to lead-time reduction in the production industry, however, the overall processes still consisted of a sequential chain of events that occurred with minimal interaction between different life-cycle phases such as design and manufacturing. Such a sequential process is often referred to as the traditional *over-the-wall* approach [Sobek, *et al.*, (1999), Womack *et al.*, (1991)]. Several automotive manufacturers such as Toyota and Ford were quick to realise that such an *over-the-wall* approach due to its inherent long lead times could endanger their long-term competitiveness and limit their ability to develop better quality products[1]. During the 1970s, major automotive manufacturers were already outsourcing a considerable amount of jobs (eg., milling, painting, moulding) to reduce cost. However, it needed intense collaboration between partners over various stages of product development [Nishiguichi(1994)].

To overcome the drawbacks suffered by the sequential engineering approach, a new methodology called *concurrent engineering* (CE) emerged during the 1980's which relied on parallel activities in the product development process [Ainslie, *et al.*, (1985), Walkett, (1989)]. The main requirement of concurrent engineering (CE) was on collaboration and group decision making. CE drew immediate attention of a number of manufacturers due to its emphasis on lead time reductions, quality improvements, scrap reduction and cost reduction [Poolton and Barclay, (1996)]. Subsequent developments in computer based tools for Computer Aided Engineering (CAE) [Ainslie, *et al.*,], Design for Manufacture and Assembly (DFMA) [Boothroyd (1988)] and Quality Function Deployment (QFD) [Akao, (1990)] encouraged the integration of disparate phases of product development. Additionally, development in production methodologies such as agile manufac-

[1] As pointed out by Smith and Eppinger (1997) modern engineering views iterative product development process as largely inefficient excluding situations where optimal sequencing between product development phases are inevitable due to inter-dependencies.

turing [Meier, *et al.*, (1994)], and the concept of reconfigurable and modular manufacturing cells [Alford, (1994)] readily suited the CE philosophy of multi-function teams and empowerment. Although CE philosophy has not yet been fully realised in industry [Waterson, *et al.*, (1999)], certain aspects of CE such as specialist team formation and information sharing are now well supported and implemented for 88% of the UK industries, as reported by Theobald (1996).

1.2 Motivation

The spread of global computer networks during the 1990's has been influencing every dimension of product life cycle from marketing to design, manufacturing and distribution [Bentley, *et al.*, (1994), Sobolewski and Erkes (1995), Dietz (1997) Hameri and Nithila (1997)]. Product development processes, whether sequential, concurrent or a combination, need to be flexible enough to allow collaboration between partners that are geographically distributed. This is due to the growing trend towards globalisation in the engineering industry during the last two decades. In addition to this is the intense pressure from the customers and the need to develop quality products that could penetrate international markets in minimum time. This has led to the concept of distributed product development which is captured in the following statement:

> *"As a result of increasing global competitive pressures, corporations in every industrial sector are addressing product development challenges by focusing on what each does best and turning into partnerships to supplement internal abilities. In the manufacturing sector, this trend is manifested through joint manufacturing ventures, increased outsourcing and supplier partnerships, often with geographically distributed partners"* (Erkes *et al.*, 1996)

In a distributed environment, partners with complementary competencies collaborate to design, manufacture and market a product. These partners who may be geographically separated, virtually co-locate over wide area networks (WAN). This virtual co-location leads to a distributed organisation which is also referred to as a *Virtual Enterprise (VE)*. The potential of such organisations is yet to be fully realised in creating new business ventures. Byrnes (1993) pointed out that such a distributed environment must offer competence and trust with no formal boundaries. The long term success of distributed organisations largely depends on their ability to collaborate in order to exploit the market opportunities in innovative ways while adapting to market and technological changes in nearly a real-time manner.

1.3 Collaborative product development

The focus of the next generation product development is on developing products in collaboration using the expertise and facilities that are distributed over WANs. Researchers have proposed collaborative approaches based on intelligent software agents [Gensereth and Ketchpel (1994)] and Computer Supported Cooperative Work [CSCW] (Grudin, 1994), collectively referred to as Collaboration Technologies (CT). The former approach focuses on synthesising specialised software tools which are characterised by their autonomous decision making capability [Ndumu and Nwana, 1997]. The latter relies on real-time co-location of humans into workgroups using multi-media communication tools for collaborative decision making [Grudin (1994) and Klein (1990)]. Both approaches have received significant attention in recent years regarding their potential application in collaborative product development.

1.3.1 CSCW based collaboration

The research in office automation in the 1980's led to the concept of Computer Supported Cooperative Work (CSCW) [Hirschheim (1985)]. Initially the approach was meant to bring communication facilities such as fax and telephone to support group decision making and informal negotiation. With rapid advances in communication in the last two decades, the present day CSCW focuses on real-time virtual co-location of humans to support team work. Cutkosky, *et al.,* (1996) investigated sharing of product data and function over wide area networks. Their investigation led to the conclusion that it was time efficient to exchange product related drawings by fax and have the recreated by humans at a different geographical location than to use data exchange protocols such as Initial Graphics Exchange Specification (IGES) that may not ensure cross compatability at all times. Some of the efforts of Grudin (1994), Reinhard, *et al.,* (1994) and recently by Schiffner (1998) were on integrating attributes of face-to-face communication such as eye contact, gestures and voice with off-the-shelf desktop tools such as MBone with shared whiteboard.

In CSCW-based collaboration, the tasks of decision making and reaching group consensus have been largely left for humans to resolve. The conflict resolution expertise required to reach consensus resides in the humans themselves in the form of establishment of priorities, previous experience in group decision making and cognitive skills for bargaining. This according to Johansson, *et al.,* (1994) could increase the mental work load for humans and could be a cause for irrational decisions. This suggests that there is a need for integrating tools for rational decision making and conflict resolution in a CSCW based collaborative process. It can be argued that the software agent based approaches could complement a CSCW based process by providing such tools.

1.3.2 Agent supported collaboration

Agent supported computing is a relatively new paradigm for developing intelligent software applications[2]. An agent is a computer program situated in an environment and is capable of autonomous decisions. In addition to autonomy, it should also exhibit qualities such as ability to communicate using some formal language and negotiate using some rational mechanism. An agent may also possess adaptability to orient its preferences by perceiving its own environment [Woolridge and Jennings, 1995]. These qualities make software agent-oriented techniques a useful means for collaborative problem solving, especially if the tasks are repetitive and highly specialised [Berker and Brown (1996)].

There are several instances of agent supported techniques for collaborative product development but largely the focus has been on software considerations (eg., network communication and interoperability between heterogenous systems [Hardwick *et al* (1995), Vernadat (1996a)]. There is, however, very little information on the agent's synthesis techniques applied to engineering problem solving. In particular, the synthesis techniques need to be explicit to study how the agents generate their preferences. For example, work by Balasubramanium and Norrie (1996), Nwana and Woolridge (1997), Ndumu and Nwana (1996, 1997) and Boden (1994) provided merely a theoretical description with little evidence to support the validity of their techniques.

Since the agents have their own preferences on a common design goal, a multi-agent system often manifests conflicts between the agents. These conflicts need to be resolved and their resolution would ideally require formal reasoning and negotiation procedures. Agent supported conflict resolution has received great attention in non-engineering e-commerce applications (eg., buyer-seller scenarios of bargaining) as reported by Oliver (1996). In engineering, agent supported conflict resolution has only been treated as a domain specific issue [Berker and

[2]The term *agent* is used to imply a *software agent* unless stated otherwise.

Brown (1996)] and there are no known schemes for domain-independent conflict resolution in collaborative design and manufacturing. Even here, there is little evidence of a formal mechanism being applied for agent based reasoning and negotiation although some protocols do exist outside agent based research, for example, see Badrinath and Rao (1996), Kusiak and Wang (1996), Oh and Sharpe (1995) and Kanappan and Marshek (1993). Moreover, the existing literature in agent supported conflict resolution tends to be abstract with little emphasis on the agent's actual negotiation protocol or even on negotiation techniques [for example, Dunskus *et al*, (1995), Berker and Brown (1996)].

1.4 Conflicts in collaborative product development

The techniques discussed in Section 1.3 offer solutions covering individual aspects of collaborative design and manufacturing (e.g., communication, problem solving and conflict resolution). Specifically, the discussion highlighted the traditional human-centred approaches and certain problem-specific agent based techniques to support collaborative product development.

Pahl and Beitz (1996) have divided the design process into three main stages: conceptual, embodiment and detailed design. These are followed by the physical realisation of the product. During these stages of product development, the agents need to collaborate to make decisions on relevant aspects of the product and its processes. The conceptual design stage involves the generation and selection of the conceptual solutions for a design problem. The formulation of these solutions is based on the functions of the product, leading to a working structure of the product at the conceptual level. Any differences between designers on the formulation of solutions at this level are resolved via informal face-to-face discussions based on some technical or economic criteria. Collaboration between the

designers at this stage can be facilitated using the CSCW approach.

During the embodiment design, the individual sub-problems are identified and physical solutions to support the design concept are determined. Solutions for each sub-problem, depending on its complexity and nature of solution, can be carried out by one or more designers. Any differences between the designers will be at the schematic level eg., about sketches and flow charts. Collaboration at this stage can still be supported using CSCW approach.

The detailed design stage often involves the deployment of computer-based tools such as CAD/CAM software, optimisation programs and engineering analysis tools. Increasingly, such computer based tools are capable of decision making in their specific area of expertise and tend to have a resemblance with software agents. Design conflicts at this stage should be treated as a collaboration issue between agents.

To address the problem of collaboration between agents in collaborative product development, agent synthesis and conflict resolution need to be addressed as an inter-related issue. The reason for this inter-relation is that the ability for automated conflict resolution will depend on the underlying decision mechanism used by the agents. Therefore, an agent model should be synthesised with an appropriate decision mechanism that will be conducive to conflict resolution. The agent's preferences will need to be captured in the agent model by some formal means to represent the autonomous behaviour. The conflicts between agent preferences will need to be resolved using some rational scheme.

The following points emerge from this discussion:

- Need for an approach for software agent synthesis;
- Need for a formal scheme for conflict resolution between agents.

Based on the preceding discussion, the objectives of this research are identified next.

1.5 Research objectives

The aim of this research is to enhance the computer-based techniques for collaborative product development. Specifically, the research has the following objectives to support the detailed design:

Objective 1: *To develop an effective technique for agent synthesis to support collaborative design;*

Objective 2: *To develop a novel scheme for conflict resolution between agents.*

1.6 Research methodology

The methodology employed to achieve the objectives of this research consists of the following steps:

1. Review of previous research related to collaborative product development and identify key outstanding research issues.
2. Analyse existing approaches for modelling software agents and adopt or modify these for constructing software agents to collaborative engineering.
3. Development of a scheme for conflict resolution in an agent supported product development process.

4. Validation of the new software agent synthesis approach and conflict resolution scheme by means of laboratory-based design and manufacturing case studies.

1.7 Thesis outline

This thesis on conflict resolution in a agent supported product development process is organised as follows.

Chapter 2 presents a review of literature related to collaboration and conflict resolution based on intelligent software agents and Computer Supported Cooperative Work (CSCW); the area of software agents including agent synthesis is explored in detail. This review covers agent related research based on its relevance to conflict resolution in design and manufacturing. The existing conflict resolution schemes are reviewed and the outstanding research issues are identified in accordance with the aims of this research.

In Chapter 3, a novel technique is proposed for synthesising Multi-Agents. This technique is based on the concept of Single Function Agents (SiFAs) which are modelled using the Habermas's theory of action. Internal structure of the new SiFA model is discussed. The chapter concludes with a description and two examples of how single function agents can be combined to form multi-agents.

Chapter 4 presents a novel scheme for conflict resolution in a multi-agent decision process. First a background is presented on conflict resolution in product development. Then two key approaches for conflict resolution namely constraint relaxation and game-theoretic negotiation techniques are discussed and a need for a unified scheme is identified. A new conflict resolution scheme developed in this thesis is presented in detail. The underlying reasoning and negotiation agents have been explained to show how these could be used to assist rational

decision making.

Chapter 5 presents a comparative case study aimed at benchmarking the new conflict resolution scheme by applying it to the collaborative design of a poppet relief valve. Comparison with previous research work in conflict resolution is made on the basis of this case study.

Chapter 6 presents a case study on the design and manufacture of a Geneva mechanism to validate the agent synthesis and conflict resolution scheme in a wider collaborative process consisting of software agents and human decision makers.

In Chapter 7, conclusions of this research are drawn and comparisons made with the other work in collaborative product development in the context of conflict resolution. The key contributions to knowledge and understanding of the subject area of conflict resolution using software agents are highlighted. Finally, limitations of this thesis and scope for further research are identified.

Chapter 2

Review of agent synthesis and conflict resolution

2.1 Introduction

This chapter presents a review of previous research relating to software agent synthesis and conflict resolution between agents in collaborative product development. The review focuses on the issues related to the objectives of this research, identified in Chapter 1. The literature review is intended to provide the reader with necessary background for the rest of this thesis. Accordingly, this chapter is organised around the following aspects:

- A background to the trends in collaborative product development is presented. This will give an overall picture of collaborative product development based on the approaches of Computer Supported Cooperative Work (CSCW) and software agents.
- A detailed review of software agent based technique for collaborative product development is presented. This is to highlight some key characteristics

of agents and the application of agents in design and manufacturing.

- A review of agent synthesis: Single Function Agent (SiFA) based approach for synthesising software agents is presented and a comparison is drawn with Multi-Agent approach.

- Existing schemes for conflict resolution relating to collaborative design are reviewed. A need for an explicit scheme for the resolution of conflicts between software agents is identified.

- Identification of relevant research issues and formation of two hypothesis for this research.

This review has drawn on various sources including journal articles, conference proceedings, text-books and magazines. In addition, it also includes a few private communications between the author and experts from the academia and industry.

2.2 Collaborative approaches

In collaborative product development partners with complementary competencies virtually co-locate to design, manufacture and market a product. Brynes (1993) was one of the first authors to state the attributes of a collaborative distributed organisation. According to Brynes, a distributed environment involves seamless integration across partnerships without formal boundaries. This implies that extensive infrastructure support of collaboration technologies (CT) would be necessary to overcome barriers such as geographical separation. The CT has been broadly identified to be comprising of both hardware and software, the specific details of which will depend on the environment.

There has been extensive research since the 1980's in the areas surrounding collaborative work [for example, see Hirschheim (1985)]. Two key techniques emerged

out of this extensive research, namely CSCW and software agents. Though the common goal of these techniques was to support collaborative work, their evolution has been independent. In the context of distributed product development, these techniques address a wide range of issues highlighted in Chapter 1 such as communication, interoperability and decision making. In what will follow, an overview of CSCW and software agent based approaches is presented.

2.2.1 CSCW based approaches

Research in group working and office automation in the 80's led to the concept of Computer Supported Cooperative Work (CSCW)[1] [Hirschhiem (1985)]. In its early stages CSCW was mainly intended towards integrating single-user applications to support desktop working. The CSCW-based research has employed video conferencing technology to assist human-centred virtual business meetings [Ishii, *et al.*, (1994)], collaborative design and manufacturing [Gay and Lentini (1995), Schiffner (1998)]. Humans orchestrate all major decisions in a CSCW-based setting and software tools are only viewed as vehicles to assist the collaborative work. This enhances the sense of control and trust between team members [Talbert, (1997)].

Though Computer Supported Cooperative Work (CSCW) has found success in several applications (e.g., distributed manufacturing), it suffers from various limitations. Vast amount of CSCW literature covered mainly the tools (eg., InPerson for video conferencing, email, telephone and fax) that assist collaboration rather than how effectively they assisted the process. For example, the studies by Ishii, *et al.*, (1994) reflect this where the focus was on providing a set of tools for collaborative work rather than on issues such as resolution of conflicts between humans.

[1] CSCW is also known as *Groupware*, *Workflow* and *Group Decision Support Systems* [Grudin, (1994)].

Increase in mental workload is seen as one of the factors that tend to diminish human performance and be a cause for irrational decisions [Johansson, *et al.,* (1994)]. The humans in a collaborative design process have individual preferences towards a common design attribute (e.g., strength, cost) which often lead to conflicts. These conflicts need to be resolved based on some formal approach rather than by some ad hoc means where the conflict resolution expertise rests on humans themselves. Thus there is a need for tools to assist humans in collaborative problem solving in the context of maintaining rational decisions and resolving conflicts. The research in the area of software agents could complement CSCW with respect to providing such tools.

2.2.2 Software agent-supported approaches

The software agents came into existence in the 1990's but their roots date back to distributed AI research in the 1980's where the work by Gasser (1988), investigated ways of building functioning, automated problem solvers for specific applications [Gensereth and Ketchpel (1994)]. For example, one such outcome of research in distributed AI was Contract Net [Gasser (1988)]. This software could exchange data but lacked problem solving ability. Later in the 1990's, due to the innovations in Internet-based applications, there was a need to develop software that could ensure reusability, problem solving ability and compatibility for data exchange across heterogenous platforms [Woolridge and Jennings, (1995)]. Thus the very idea of software agent based research was to develop systems that were capable of benefiting end-users rather than a specialised group. Some of the recent work in software agent based research demonstrates their wide applicability in work-flow management [Huhns and Singh, (1994)], electronic commerce [Sandholm and Lesser, (1998)], digital libraries [Durfee, *et al.,* (1996)], negotiation in virtual markets [Oliver, (1996)] and conflict resolution in parametric design [Berker and Brown (1996)].

2.3 Software agents in collaborative product development

The objective of this section is to present some introductory aspects of the software agent technology and examine how these aspects are realised in real-world scenarios in the context of conflict resolution between agents.

2.3.1 What is a software agent ?

The objective here is to highlight those characteristics that will differentiate a software agent from an ordinary software program. Though there are several published articles on agent based approaches, there is still a lack of agreement between researchers on the characteristics that define agent-hood [Ndumu and Nwana, (1997), Lander, (1997) and Shoham, (1999)]. Shardlow (1990) was the first to define a software agent as: *"Agents are the ones that act"*. Shardlow's definition is not too dissimilar to the meaning of the term *agent* given in the Webster's comprehensive dictionary which is *"One who or that which acts or has power to act..."*. These definitions merely indicate involvement of actions associated with each agent rather than its characteristics. In general software agents should possess key functionalities such as autonomy, adaptability, ability to communicate and resolve conflicts, interoperability and pro-activeness. A more precise definition of a software agent would require an explanation concerning specific behavioural qualities such as belief, desire and emotions. In addition to this, it would require a precise definition of the actions the agent is expected to perform and goals the agent wishes to achieve. Some of the software agent characteristics commonly referred in the literature are defined next [Lander (1997), Woolridge and Jennings (1995) and Shoham (1999)].

- **Autonomy:** Autonomy gives an agent the ability to make decisions inde-

pendently with minimal human intervention. An autonomous agent may not blindly obey commands but will have the ability to modify requests made by other agents or even refuse them in a given context such as bargaining [Etzioni and Weld, (1994)].

- **Adaptability:** This enables the agents to perceive their environment such as computer network or manufacturing plant and respond to changes in a timely manner.

- **Ability to communicate:** An agent should have the ability to communicate formally using a language such as Knowledge Query Manipulation Language (KQML) with other agents to accomplish its goals. This also refers to the use of precise vocabulary in the process of communication.

- **Ability to resolve conflicts:** An agent should be able to resolve conflicts (e.g., in situations where agents have differing preferences towards a common goal) by means of some formal mechanism. This mechanism could make use of formal procedures such as game theory or informal means such as domain-specific rule-based techniques.

- **Interoperability:** This gives an agent the ability to interact with other agents in a heterogenous environment. For example, interoperability is necessary if two agents that are supported by different software and hardware platforms wish to exchange data or functions.

- **Pro-activeness:** This refers to the agent's ability to take the initiative and exhibit opportunistic behaviour in complex problem solving environments.

A given agent may not possess all the characteristics presented above but only some of them. This has resulted in agents to be classified as weak or strong [Woolridge and Jennings, (1995)]. According to Takeda, *et al.,* (1996) a strong agent would possess additional capabilities such as belief, emotions, learning ability and obligation. Any agent that does not have these capabilities is termed weak.

Figure 2.1 shows several agent-hood characteristics put forward by researchers and highlights the variety in defining agent-hood. However, there is agreement on some of the common features an agent may possess such as autonomy, communication, interoperability and ability to resolve conflicts which distinguish a software agent from an ordinary computer program. In later sections it will become obvious as to the extent to which the characteristic of conflict resolution ability is realised in agent-supported collaborative design and manufacturing.

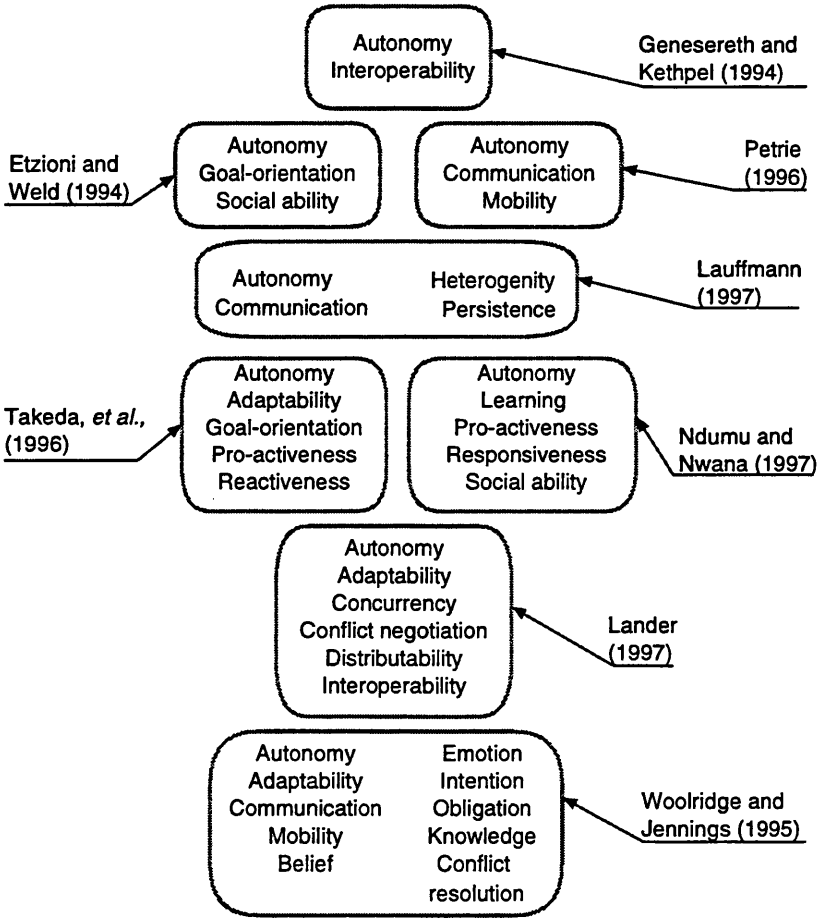


Figure 2.1: Characteristics of agent-hood.

2.3.2 Application of software agents in collaborative design and manufacturing

This section reviews key research carried out in agent-supported product development with emphasis on distributed product design and manufacturing[2]. The review is presented with respect to the extent to which the agent-characteristic of ability to resolve conflicts is realised in real world collaborative problem solving.

• McGuire, *et al.,* (1993)

This is one the initial efforts in the line of agent-based product development. Their effort was termed SHADE (Shared Dependency Engineering) and involved four geographically distributed partners - Lockheed, Stanford University, Enterprise Integration Technologies, and Hewlett-Packard. The main aim of this work was to exchange information between software agents. They addressed this using KQML to provide the means for formal exchange of knowledge expressed in KIF format, in addition to data sharing [Finin, *et al.,* (1995)]. Though their work proved the importance of communication by means of formats such as KIF, it failed to identify other agent functionalities such as negotiation which are necessary for collaborative problem solving. For example, the agents communicate using match-making, a technique similar to blackboard technology. This implies communication takes place based on the *content* field description in the KQML message structure [see Appendix C]. Their protocol applies to cases where there is only one match and did not consider a case where conflicts between agents occur due to individual preferences. In situations such as this, the agents may need to negotiate using some formal techniques such as game theory to pick a particular match. These agents demonstrate communication, interoperation and task-solving ability and hence are categorised as weak.

[2]The available literature in software agent based research is voluminous and varied. For the present research, the choice of references cited here was made based on their relevance to conflict resolution in design and manufacturing.

• Lewis, *et al.,* (1994)

Lewis, *et al.,* provided software agent services (e.g., design and manufacturing) distributed over a network (CAMnet) to enable data-sharing between geographically separated partners. Though the principal aim of this work was data-sharing, this differed from the work by McGuire, *et al.,* in two key ways. First Lewis *et al.,* provided Web-based interface to applications and did not make use of formal languages such as KQML. The sharing of product data was achieved by using exchange standards such as STandard for the Exchange of Product model data (STEP). They also provided *wrapper* mechanisms to enable data interoperability between agents but functional interoperability was not achieved. The aspect of conflict resolution between agents left out by McGuire, *et al.,* (1994) was not further addressed as a major issue. The agents presented in this study showed ability for communication, interoperation and task solving, and can be classified as weak.

• Erkes, *et al.,* (1996)

They presented an expanded effort by the CAMnet project group to enhance reuse and functional interoperability between heterogeneous agents, issues that were not covered by Lewis, *et al.,* (1994). Figure 2.2 shows the product development architecture proposed by Erkes, *et al.,* that includes networked agent services, and legacy software.

The main objective of this architecture was to maximise the automated agent-based services (e.g., design, simulation) and integration of legacy software through CORBA[3]. They demonstrated this in the context of a distributed snap-fit design problem where information exchange between a legacy FORTRAN code and a database was facilitated through CORBA. Based on their identification, their agents point towards a weak definition of agent-hood - the only capabilities identified being communication, interoperability and problem solving ability but not the ability to resolve conflicts. However, their initial success prompted them to

[3]Iona Technologies, Inc., USA. http://www.iona.com

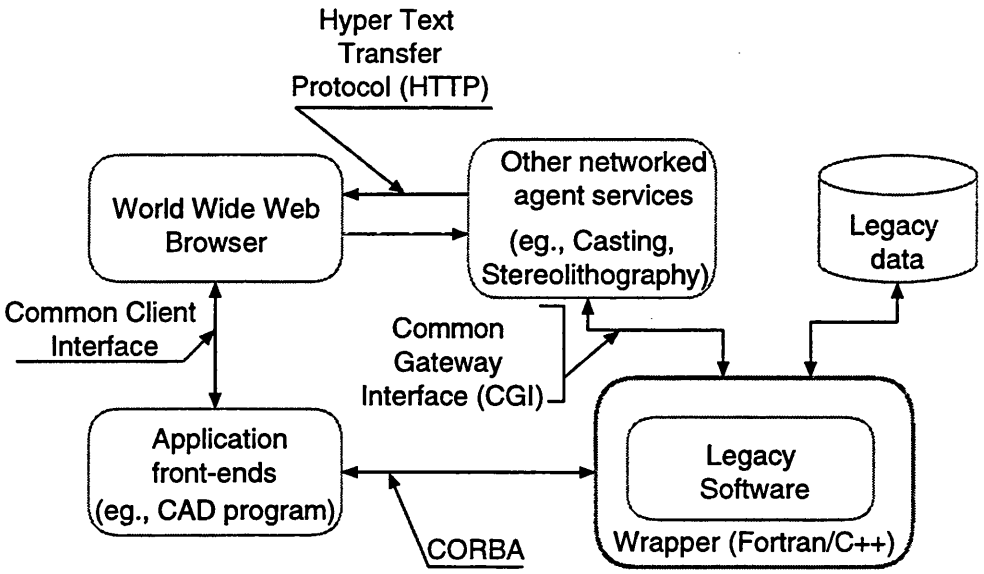


Figure 2.2: Distributed product development architecture of Erkes, *et al.*

point to certain aspects of agent-hood such as autonomy and pro-activeness which can be inferred from their statement: *"In the Object agent-model, system components dedicated to individual users are agents acting on behalf of those users"*. Careful interpretation of this statement in the context of their effort would reveal that only specific functions (e.g., automated message passing between software agents, invoking legacy software functions using Remote Procedure Call) can be executed on behalf of human users. This is due to the dearth of evidence in their work and elsewhere that would let a software agent perform other human attributes such as motivation, belief and decision making in uncertainty [For example, see Shoham (1999)].

• Yang and Ho (1994)

Yang and Ho proposed a framework for agent-based distributed product development with emphasis on negotiation and flexible open architecture. They presented a framework consisting of intelligent software agents and used terms such as *decision making, coordination and communication* that gave a theoretical description. The authors made extravagant claims, such as all processes within their framework were software agent-controlled, without giving any details of the mechanism of control. For example, the authors quote, *"...all problems (sic) solv-*

ing procedures are executed automatically. Problem requesters need not know how a problem can be solved, they only need to specify what problems they want to be solved". This statement suggests the agents actually generate solutions for the posted problems. Another statement by the authors contradicts their previous statement as, *"The artificial IA's execute tasks orderly according to the algorithms (in the problem specification base) we have given, but will not work well if undefined problems are posted".* They presented a somewhat poorly organised framework for distributed problem solving with little attention given to identifying their agent's structure, its characteristics such as negotiation and realising its merits in implementation.

• Balasubramanium and Norrie (1996)

Balasubramanium and Norrie developed an agent-based framework for concurrent product development. The central theme of their work was to agentify processes such as design, scheduling and negotiation. Much of their explanation is on theoretical aspects of product development processes such as design-shopfloor coordination. The authors present little about their agent characteristics, details of which are only limited to the imagination of the reader. This is evident in at least two instances. For example, the authors quote, *"This negotiation mechanism helps resolving conflicts among instantiated features automatically to a large extent".* No negotiation mechanism for conflict resolution or any pointers to any existing mechanisms were actually given. In another instance the authors quote, *"Every tool in the shop-floor is represented by an autonomous tool agent. Each tool agent has knowledge about its shape, schedule and tolerance capabilities in combination with a particular machine and work material..."* There was no mention of how the knowledge is represented internally (e.g., in the form of rules) or how their agent would carry out autonomous actions. Such claims without proper representation of agent-hood could at best lead to theoretical solutions and cannot be regarded as a *proof-of-concept.*

• Berker and Brown (1996)

Berker and Brown presented a Single Function Agent (SiFA) based approach for conflict resolution. They showed that using SiFAs it was easier to visualise the internal structure of agents closely (e.g., design functions at the parametric level, conflict resolution mechanisms). Their agent-model had several meta-levels of decision associated with each SiFA. However, the actual conflict resolution procedures within these levels were not stated explicitly.

Though a conceptually elegant approach, these unlimited levels of decisions to support conflict resolution made it hard to implement their agent model as they themselves acknowledged. One of the flaws of their approach, in common with that of Yang and Ho (1994), was the assumption, *"Our work uses software agents instead of human agents"* which only proved to be difficult at the end for the authors to substantiate. They emphasise on agent-characteristics such as informal communication using speech act theory and problem solving ability which place their agents in the weak category.

• Frost and Cutkosky (1996)

Frost and Cutkosky proposed a framework for distributed manufacturability evaluation. The framework encapsulated design and manufacturing software agents that communicate using KQML [Finin, *et al.,* (1995)]. While the effectiveness of KQML has been demonstrated by several others including McGuire, *et al.,* (1993), this work differed in one key respect. Frost and Cutkosky developed agent-templates for KQML-based communication in Java. Hence their agents exhibited a truly platform-independent behaviour. In addition, the agents were also encapsulated in a Web page as Java applets that enhanced portability. Frost and Cutkosky provided only generic information about agent's internal structure that consisted of design rules, constraints and simulation procedures. But their results indicated that the inclusions of simulation procedures actually enhanced the problem solving ability of agents. They demonstrated this in the context of the chatter problem detection during milling operations. There was also considerable emphasis on mapping of geometrical features (e.g., generalised cylinders)

between CAD and process planning agents thus addressing the interoperability at data and function level. However, issues such as negotiation and autonomy were not taken into consideration. Their agent-hood can at best be described as weak.

• Orbst, *et al.*, (1997)

Most of the agent-based frameworks reviewed previously covered mainly communication and interoperability. Orbst *et al.*, introduced belief and autonomy in addition to communication and task processing. They effectively captured the data exchange process in a typical setup involving a large number of agents (e.g., design, stress, manufacturing, and monitoring agents). They presented details of how the agents use rules in goal-task mapping that enable autonomy during a design process. In order to relate the autonomy of agents with their tasks, Orbst *et al.*, developed a Task-Builder tool that displayed an explicit control logic for the execution of tasks. However, they failed to bring out explictly how beliefs were modelled in their agents. In addition, Orbst, *et al.*, failed to acknowledge the existence of multiple preferences (marked by autonomy) which could often lead to conflicts between agents. Their agents demonstrated autonomy, goal-oriented behaviour, communication and interoperability which places them in the weak category.

• Zeng and Sycara (1998)

The *Bazzar* negotiation model of Zeng and Sycara (1998) presents evidence of agent based conflict resolution and learning through interactions. They presented agent negotiation from a utility point of view and examined cases: (a) when all agents learn, (b) when no agents learn and (c) when one agent learns. The effect on joint utility was also examined. Though *Bazzar* model captured the key aspects of a typical negotiation scenario (e.g., formation of utility functions, evaluation of trade-off, etc.), the focus was on learning mechanisms that support conflict resolution. Their agents demonstrated autonomy, learning capability, and ability to negotiate which places them in the weak category.

Having presented application of agents as seen by previous research, the next section will identify some of their merits and limitations in collaborative product development.

2.3.3 Merits and limitations of software agents

Realising the appropriateness of software agent technology lies in evaluating its merits and limitations applied to distributed product development. These are summarised below.

Merits

- Agent-based techniques offer efficient solutions to collaborative problem solving by hiding the complexity of the problem from other agents and at the same time delivering the solutions in a time-efficient manner. For example, the personal assistant agent developed by Takeda, *et al.*, (1996) demonstrated the problem solving ability of agents that carry out simple tasks (e.g., preparing schedules, booking venues for meetings, e-mailing project partners, etc.) on behalf of humans. Such specific problem solving ability of agents in the manufacturing domain was highlighted by Shen and Norrie (1999) where the agents prepare the schedule for manufacturing operations (e.g., assigning a machine tool, a fixture, etc.).
- Influence of agent learning in task processing [e.g., *Bazzar* model of Zeng and Sycara, (1998)] shows how learning can influence a decision process. Even though learning mechanisms are still at the development stage, these have proved to be useful in the light of some specific applications (eg., conflict negotiation, process scheduling) [Ndumu and Nwana, (1997)].

The above merits offer valid justification that software agent technology will enhance the construction of complex distributed systems given sufficient develop-

ments in agent synthesis (e.g., air traffic control, patient monitoring and spacecraft fault detection systems). However, the concept of software agents is not magical and suffers from some limitations. These are summarised next.

Limitations

- One of the major problems highlighted in Section 2.5.4 was the aspect of overselling agent technology without enough evidence on its synthesis and conflict resolution mechanisms. Most of the existing research did not focus on providing schemes for conflict resolution between agents. Some the existing conflict resolution procedures including of Berker and Brown (1996) were domain specific. Similarly, the negotiation protocol of Balasubramanium and Norrie (1996) was also not explicit with respect to the conflict resolution knowledge.

- One of the issues was replacing humans by software agents. This is clearly beyond the scope of the present technology and is well brought out by Woolridge and Jennings (1995) in the context of decision making: *"Those unfamiliar with the achievements (and failures) of Artificial Intelligence (AI) often believe that agents are capable of human-like reasoning and acting. Obviously, this is not the case: such a level of competence is well beyond state of the art in AI. Thus agents may sometimes exhibit smart problem solving behaviour, but is still very much limited by the current state of the art in machine intelligence"*. This argument was well supported by Shoham (1999) based on his comments on the current trends of software agent technology.

- Finally, there is a lack of standards in synthesising agent-based systems including standardised support for conflict resolution, tracking and monitoring processes, run-time management, etc. For example, most of the existing agent-development platforms such as JATLite and JAFMAS provide a base for KQML-based communication. By providing a standardised

support, these platforms could do more than just exchange KQML messages.

Having presented the limitations of agent-supported technology, the the SiFA based approach for the synthesis of software agents is reviewed. This review is necessary to support the first thread of this thesis which is synthesis of software agents as stated in Section 2.1.

2.4 Software agent synthesis

Research work in the area of Multi-agents (MAs) covers different domains such as information systems, task processing and mediation. These MAs are large grained; their granularity refers to the large information contained in such agents. Some of the examples of such large grained agent based systems are SHARE [Toye and Tanenbaum (1993)] and ABCDE [Balasubramanium and Norrie (1996)].

The treatment of the building of such large grained agents has been abstract in the literature [Berker and Brown (1996)]. For example, details of mechanisms for communication, problem solving procedures, and schemes for conflict resolution are generally not presented. Thus, the internal structure of the agent is opaque [for example, see Balasubramanium and Norrie (1996)]. In an attempt to study these agents at a finer level of granularity, Dunskus, *et al.,* (1995) proposed an approach based on Single Function Agents (SiFAs).

2.4.1 Background on agent synthesis

Single Function Agents (SiFAs) are a way of constructing multi-agent systems. A single function agent is meant to achieve a single objective based on a single target

which is a design variable [Figure 2.3]. In a problem solving process the SiFAs interact with other agents based on a pre-defined strategy. A strategy is usually a number of rules and constraints that guide an agent into taking appropriate actions during various stages of problem solving.

The SiFAs contain limited knowledge for task solving and conflict negotiation [Shakeri (1998)]. This knowledge is identified as design knowledge and conflict resolution knowledge. The design knowledge relates to the design variables, and engineering design methods whereas the conflict resolution knowledge includes strategies such as game theoretic techniques.

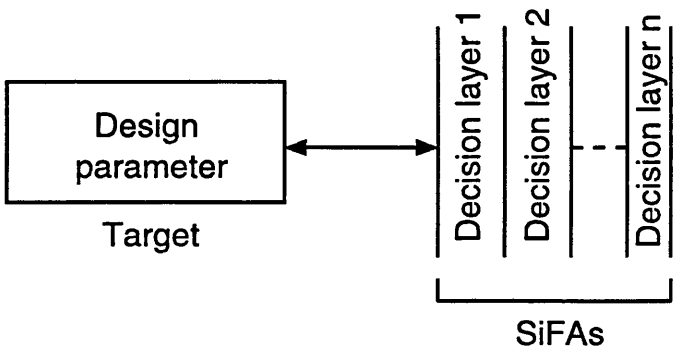


Figure 2.3: Levels of decision in Dunskus, *et al.*, (1995) and Berker and Brown (1996)

Background work on SiFAs is rather limited and not as extensive as work on multi-agent systems. SNEAKERS was the first SiFA-based system developed by Brown (1992). The system consisted of several task-specific SiFAs that assisted the user by suggesting and criticising solutions to a design problem. Brown did not, however, demonstrate the applicability of SNEAKERS to a case study. Victor, *et al.*, (1993) used a SiFA based approach, known as I3D, a parametric design system. Agents in I3D was responsible for individual tasks of design evaluation, cost estimation and manufacturing. The I3D system did not have a conflict resolution mechanism as the choice of design variables was fixed at the start of the design process. One key enhancement from Brown's work was that Victor *et al.*, practically demonstrated the SiFA approach in the context of the design and manufacture of ceramic components.

Dunskus, *et al.* (1995) presented the SINE project and a theoretical overview of SiFAs. For example, they discussed interaction patterns of exchange information between SiFAs. However, their work remained largely generic and was not evaluated from the perspective of its practicality and suffered from the lack of case studies. As shown in Figure 2.3, the proposed SiFA model had several decision levels that remained too abstract to be defined. Berker and Brown (1996) refined the approach of Dunskus *et al.,* in terms of identifying certain types of SiFAs (e.g., suggestors, advisors, etc.). Though Berker and Brown presented an agent model based on SiFA, they placed little emphasis on: (a) explaining the formal procedures their agents would adopt for problem solving, (b) conflict resolution schemes and (c) communication mechanism. These aspects remained largely generic with some minimal information presented in the context of the design of a wine glass. They also did not relate how SiFAs combine to form MAs. Before attempting to visualise the transition from SiFAs to MAs, it is necessary to see their distinguishing characteristics.

2.4.2 Differences between Multi-Agents and SiFAs

From the view point of developing software agents, it is important to underline the distinction between SiFAs and Multi-agents (MAs). Table 2.1 presents a set of four criteria that compares SiFAs with MAs.

A Multi-agent system often has several associated perspectives (e.g., design, manufacturing) whereas SiFAs deal with a specific design perspective such as compressive strength. Creation and maintenance of ontologies are simple in the case of SiFAs as they refer to a particular design attribute. In the case of Multi-agents, making compatible ontologies is often difficult due to the heterogeneity of agents and perspectives. Thus communication support will need to be robust (by means of effective capturing of ontologies) in a Multi-agent environment. Agent behaviour is more predictable at the attribute level as in the case of SiFAs (e.g.,

Table 2.1 Comparison of Multi-agents and SiFAs

Criteria	Multi-agent systems	Single Function Agents (SiFAs)
Perspectives	Influenced by several perspectives (e.g., design, styling)	Limited perspectives (e.g., stress function)
Ontology	Developing compatible ontologies between perspectives is difficult	Ontologies in this case refer to a simple design attribute
Communication	Communication may require special protocols such as CORBA	Communication is simple and involves sharing of parameters
Behaviour formulation	Individual preferences are often non-deterministic and fuzzy (e.g., colour)	Behaviour is predictable and is often well defined (e.g., mathematical functions)

evaluation of stress, etc.). In Multi-agent systems, individual preferences exist; these often lead to conflicts. These conflicts often occur at a higher level of abstraction and are often fuzzy, for example, preferences on appearance or colour of a product [Kusiak and Wang, (1995)]. As highlighted in Section 2.3.5, there are no known schemes for resolution of conflicts between agents in a domain-independent manner. With the objective of developing such a scheme a review of conflicts and available schemes relating to collaborative design is presented next.

2.5 Conflict resolution between agents

The literature relating to conflict resolution is voluminous and varied, embracing contexts such as business, jurisprudence, international relations and engineering [Nash (1950), Binmore (1985), Klien (1993) and Rao and Freheit (1991)]. In this section a subset of this literature relating to computational models for conflict resolution relating to parametric design is reviewed. The key aim of this review is to uncover the conflict negotiation expertise applied to collaborative design and manufacturing involving both software and human agents.

2.5.1 What is a conflict?

There is not yet a universal definition for the term *conflict*. According to Rahim (1986), a conflict occurs when there is a disagreement between two objects. The cause for this disagreement could be due to several factors such as objects holding different viewpoints towards a decision variable or the evaluated decision variable fails to satisfy one of the objects' interests. March and Simon (1993) defined conflict as a situation where a group experiences difficulty in choosing alternative actions due to a breakdown in the decision making mechanism. Berker and Brown (1996) viewed conflicts as an occurrence in a design process where one or more design constraints are violated. A similar view was also adopted by Kusiak and Wang (1996, 1995) and Wang (1994). Most definitions of conflict seem to centre around the argument that there is some dissimilarity between two agents towards a common goal [Nash (1950), Rahim (1986), Kanappan and Marshek (1993), Rao and Freheit (1991), Wellman (1990)][4]. These preceding characterisation of conflict have been modelled and captured in discrete symbolic and mathematical forms (e.g., truth maintenence, AI-based reasoning and game theory) and have been applied to group decision situations in collaborative design. For example, Medland (1992) developed constraint-based modelling procedures for collaborative design. These rule-based procedures are encapsulated in a modelling software[5] which conducts a search to determine a state where the rules are true. Such modelling techniques are valuable in contexts where interdependencies between product life-cycle processes exist [Gonikhin and Medland (1990)].

Having presented the different views of conflicts, the author has chosen to define a conflict in parametric design as: *A collaborative decision situation where two*

[4]The discussion here is restricted to the definitions of the term *conflict* put forward by researchers mainly in the context of collaborative design. A more discursive treatment on classical vs. non-classical conflicts (e.g., international business negotiation, strategic warfare, etc.) is not presented here as it is beyond the scope of this work.

[5]CAMFORD: Reference manual, Constraint modelling collaborative group, Department of mechanical engineering, University of Bath, Bath BA2 7AY, UK.

or more agents with individual preferences disagree on a mutually shared design goal. It is this mis-alignment of multiple preferences that lead to conflicts between agents. A generic taxonomy is presented next which will identify the conflicts due to individual preferences in a wide spectrum of conflicts.

2.5.2 A generic taxonomy of conflicts

Klien (1993) presented a generic taxonomy of conflicts encountered in collaborative design. Klien's taxonomy is adapted and extended in Figure 2.4 which shows several levels of conflicts that are generally encountered during a product development process. Certain types of conflicts presented in this taxonomy, as identified by shaded regions in Figure 2.4, were identified by Klien. This taxonomy divides conflicts into three major types: (a) hard conflicts, (b) medium conflicts and (c) soft conflicts. The reason for this classification is that every conflict resolution needs a unique combination of different techniques and allowable resolution time. For example, the resolution of conflicts arising due to a lack of semantics may need less time and resources as compared to the ones arising due to differing preferences.

Hard conflicts

Hard conflicts are perhaps the most difficult to resolve since the conflicting agents are *not always in a hurry* to reach a compromise. Jelassi and Foroughi (1989) have considered some conflicts which are difficult to resolve as a *win or lose game.* This type encompasses a variety of conflicts ranging from constraint violation in product design to socio-cultural differences in decision making. Key stages in approaching hard conflicts involve identification of a conflict and applying some standard negotiation schemes such as game theoretic approaches to generate a trade-off between agents. For example, in a pressure vessel design problem, an

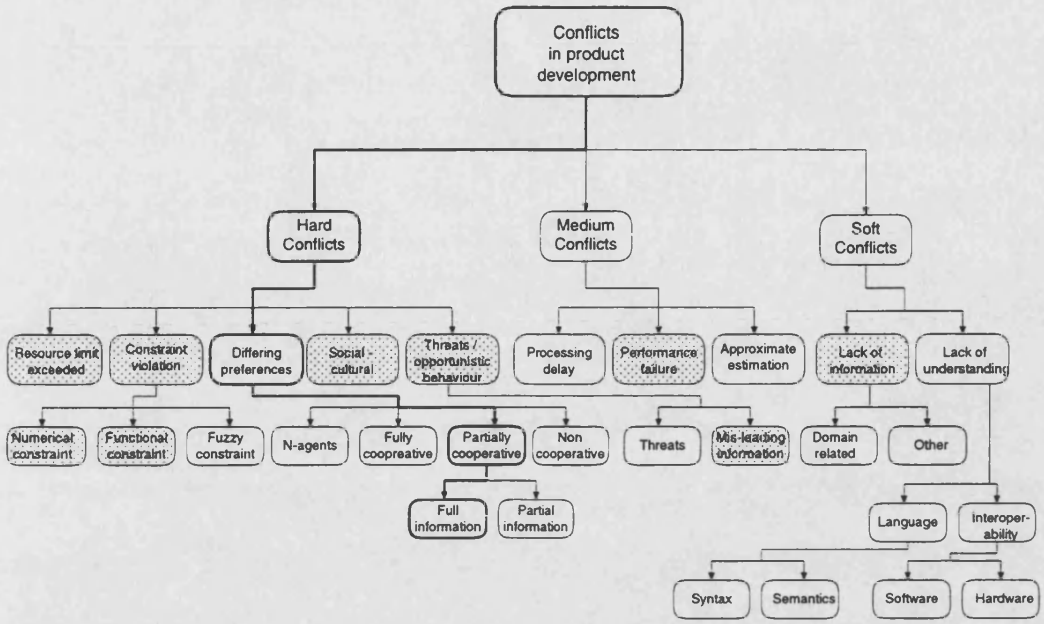


Figure 2.4: A taxonomy of conflicts

agent's objective might be to minimise the weight whereas another agent's interest may be to maximise the volume [Rao, *et al.*, (1997), Lewis and Mistree (1998)]. Such a scenario where there is a set of preferences for each individual agent on shared or interacting parameters could lead to a hard conflict.

In a different case, an agent's preference may be on a certain fuzzy characteristic such as the colour of a product. In such cases a fuzzy logic based approach of Kusiak and Wang (1995) could be applied to enforce a settlement between conflicting agents. Researchers have focused on many such schemes for negotiation but still a wide spectrum of hard conflicts remains unexplored [e.g., n-agent negotiation, threats in bargaining]. Many of the traditional approaches based on constraint relaxation, game theory and MCDM techniques assume that the preferences of agents remain constant. This assumption is often flawed when agents wish to change their preferences in a dynamic problem solving environment and thereby give rise to secondary conflicts. As Vincent (1981) stated, such decision making under uncertainty is quite complex and a generic solution covering all aspects may be almost impossible.

Medium conflicts

Medium conflicts arise mainly due to approximate estimation, performance failure or any other unexpected delays (e.g., caused due to accidents) in the product development process. This type of conflict often requires what is called *approximate cooperation* [Lewis and Mistree (1998)]. As Ward (1995) pointed out, at Toyota, the designers often deliberately come up with approximate specifications initially when interacting with suppliers. This is done in order to save time and resources at the start of the design process. However, such an initial specification could lead to a conflict between feasible design values for a product. One of the key approaches to resolve such conflicts is to create approximate expressions for decision variables in the absence of full information. Mathematically this is accomplished by using a Taylor's series expansion of first or higher order terms of the agent-controlled variables [Hacker and Lewis (1998)]. Other instances of medium conflicts are due to performance failure where the intervention of automated design experts is often suggested [Klien and Lu (1995)]. In this case, machine learning techniques such as case-based reasoning could be applied in conjunction with an automated negotiation system [Sycara and Lewis (1991)].

Soft conflicts

Soft conflicts usually occur between agents in a communication process. These often occur due to semantic mis-match or a lack of information that is easy to rectify. Though no rigorous procedures are adopted for the resolution of soft conflicts, they still need to be resolved in a timely manner. Klien and Lu (1995) and Berker and Brown (1996) suggested the use of speech act theory based approaches. The schemes include formal communication and use of formal languages such as KQML.

Given the three classes of conflicts, a conflict situation is rarely pure. There could

be cases involving a hard conflict on parametric values, a medium conflict due to approximate specification and soft conflicts due to imprecise communication, thus requiring a combined negotiation strategy for conflict resolution. Discussion of such strategies in the author's research will be centred around two distinct approaches based on game theory and dependency based reasoning which have been applied in the resolution of hard conflicts.

2.5.3 Resolution of hard conflicts

Conflicts are inevitable in any design process. Conflict resolution has been studied in a variety of industrial cases ranging from human conflicts (Nash, 1950) to product design conflicts. Researchers including Nash, and Klien (1990) have sought to build models for the resolution of conflicts to gain deeper understanding of issues that affect negotiation outcomes. Unfortunately, adequate models for conflict resolution do not exist except for certain polar cases of perfect conflicts[6]. For example, Rangaswamy and Shell (1997) reported a low success rate in their international business negotiation experiments (of which only 4 out of 34 cases reached consensus) even with an explicitly imposed preference structure. They also reported the non-deterministic nature of the process as marked by systematic deviations from rational behaviour. Oliver (1996) stated the need for computer-based models to improve the quality of negotiated settlements by providing means for enforcing rational behaviour. As opposed to such broad based conflict scenarios presented by Rangaswamy and Shell, parametric design processes offer a consolidated platform to analyse the conflicts in a more deterministic manner.

As the design process evolves it is frequently confronted with problems from various perspectives (e.g., design functions, manufacturing and maintenance, etc.) These conflicts could occur between humans and software agents. Detection of

[6]A perfect conflict depicts a situation where the conflicting agents opt for fair bargaining without the presence threats or opportunistic behaviour

conflicts at an early stage is essential as it directly affects the product development time. For the resolution of conflicts in a design process, there is a variety of techniques broadly identified:

- Group decision support and rule based techniques[7]
- Game theory based techniques
- Dependency based constraint relaxation

(a) Group decision support and rule-based schemes

Klien (1993)

Klien (1993) and later Klien and Lu (1995) suggested the use of a generic scheme for domain independent conflict resolution based on two key tenets: explicit capturing and efficient organising of conflict resolution expertise. The conflict resolution expertise is represented in the form of meta-knowledge and the conflicts are arranged as an abstract class hierarchy. Klien's negotiation model differed from the traditional *over-the-wall* approach as it was aimed at enhancing the *co-operative design spirit* by means of well coordinated interaction between human agents. While Klien presented a generic class of conflicts (identified in the context of a building design problem) the actual schemes necessary to support the negotiation process were not presented. For example, a part of Klien's model was aimed at studying conflicts between humans during a group decision situation. Some of the conflict resolution expertise resided in the humans themselves and the nature of such expertise was not explicitly stated.

[7]There are formal as well as informal rule-based constraint relaxation techniques. For example, Berker and Brown (1996) presented a rule-based approach for conflict resolution. These rules were based on speech act theory which use statements such as *Agent A disagrees with Agent B* without explicit reference to the object of disagreement. This approach is highly informal and lacks genericness compared to the formal truth maintenance based approach of Medland (1992) and that of Kusiak and Wang (1995).

Berker and Brown (1996)

Berker and Brown (1996) and previously Dunskus, *et al.*, (1995) presented a Single Function Agent (SiFA) based informal approach for conflict resolution between software agents. They identified a type of conflict due to the violation of design constraints and presented some informal schemes based on constraint relaxation. This, however, is based on some arbitrary variation of violating constraints and not according to dependencies between design parameters. Chapter 4 also highlighted the lack of generic negotiation strategies in their SiFA based model. For example, their work lacked focus on preference-led conflicts which is of importance in a group problem solving situation.

(b) Game theory based techniques

Kanappan and Marshek (1993)

Kanappan and Marshek studied conflict negotiation applied to parametric design problems, specifically to the design of a poppet relief valve. Their negotiation protocol was based on the following:

- Conflicts always occur between two agents;
- Preferences of agents are stated as utility functions;
- Negotiation is based on Nash and Kalai-Somordinsky techniques.

Well defined analytical formulae were used to represent the objective of each agent. In addition, the agents' preferences over a parameter were restrained from change during a design process. As two agents are involved in a design scenario, conflicts due to differing preferences are detected on variables that are shared between the two agents. The agents use game theory based techniques for conflict settlement. However, their study excluded the following aspects:

- Inter-dependencies between design variables;
- Final negotiated single-point solutions;
- Global consistency of design solutions.

Parametric dependencies do exist and their effect needs to be propagated throughout the design domain in order to obtain consistent solutions. Though work of Rao (1987) prior to Kanappan and Marshek demonstrated the use of game theory based techniques for resolving design conflicts, the former used mathematical objective functions directly relating to the conflict parameter whereas the latter used a utility based approach which represents an index (or ranking) of the conflict variable [8]. At this point it is interesting to note the view that Raiffa [Raiffa, 1972], one of the key contributors to the development of game theory, once expressed: *"I never really used the techniques of game theory - concepts and ideas, yes, but techniques, no - in my roles as negotiator....the qualitative framework of thought was repeatedly helpful - not its detailed, esoteric, quantitative aspects. Simple back-of-the envelope analysis was all that seemed appropriate"*. Rothkopf and Hanstad (1972) also echoed this view of Raiffa. However, this view is no longer true due to the recent developments in the areas of machine learning, automated decision support and electronic commerce which have been fostering many changes to the landscape of collaborative product development.

Several other game theory based protocols presented in the literature including that of Rao and Freheit (1991), Badrinath and Rao (1996), Lewis and Mistree (1998) have applied existing techniques such as Nash and Stackelberg games to human centred conflict resolution. They did not, however, consider the effect of parametric dependency or provide means for isolating a final negotiated solution.

Sycara and Lewis (1991)

Sycara and Lewis (1991) studied conflict resolution in co-operative design using

[8] The utility based approach is often preferred in circumstances where non-determinate functions could exist, for example, see Nash (1950).

Case-Based Reasoning (CBR) and qualitative assessment of strategies. CBR was used with a view to making the conflict resolution process dynamic, as agents' case base is constantly enriched with new experiences that could be reused at a later stage. Their approach allows change of preferences while negotiation process is on going, an aspect that is not present in the work of Kanappan and Marshek (1993). The approach of Sycara and Lewis is often referred to as an *aspiration* based approach and has been applied in the context of the group decision situation by Kersten (1988) and Davey and Olson (1998).

(c) Constraint relaxation techniques

Kusiak and Wang (1995, 1996)

Kusiak and Wang approached conflict resolution based on qualitative and quantitative reasoning in a constraint network representing a design perspective such as manufacturability. Each perspective has several sub-problems which are described by analytical formulae. The conflicts that occur within a perspective are studied on the basis of the violation of design constraints. The aspiration based approach of Kersten (1988) was adopted to represent the preferences of each perspective and sub-problems. Kusiak and Wang demonstrated the importance of consistency maintenence of solutions applied to the design of a poppet relief valve. However, they failed to capture explicitly the role of designers in a negotiation process. More importantly, their iterative constraint relaxation process was not shown to converge in a general situation. For example, Kusiak and Wang assumed certain relaxation levels for the conflicting parameter controlled by different perspectives (e.g., design and manufacturing). By relaxing the constraints belonging to conflicting perspectives, it cannot be stated unconditionally that their approach would converge to a final settlement of the conflict. Moreover, constraint relaxation as proposed by Kusiak and Wang presented a case where each point within a feasible design space represents a negotiated solution [Wang (1999)]. This could lead to secondary conflicts (in cases where preferences exist)

and hence such a possibility should be avoided.

2.5.4 Relative merits of conflict resolution schemes

The game theory, constraint reasoning and other rule-based techniques offer certain prescriptive solutions when applied in isolation but however, suffer from certain limitations as identified in Figure 2.5. In order for agents (both human and software) to negotiate effectively, they should be aware of parametric dependencies as well as the importance of a single point solution (as seen in game theory).

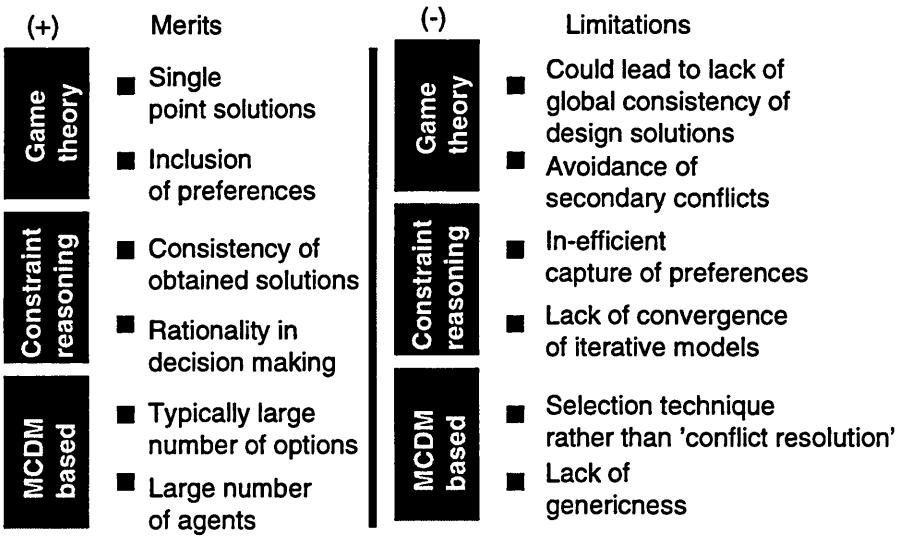


Figure 2.5: Key merits and limitations of the conflict resolution approaches

The following key points are summarised from the above review:

- Despite the success of certain prescriptive solutions for conflicts resolution, the previous approaches remain mutually isolated and have failed to coalesce into a coherent scheme for negotiation;
- Whereas the constraint-based approach give a feasible solution space, game-theoretic approaches give single point solutions;

- None of the implemented negotiation protocols combine game theory with dependency based constraint relaxation in the context of conflict resolution. Such a combination in the author's view could assist in maintaining rationality in a preference based decision process;
- Most of the techniques do not provide additional mechanisms for isolating a solution from a feasible set;
- Need for humans as an explicit aid for conflict resolution under uncertainty.

Thus there has been no known attempt prior to the author's work that directly addresses the third hypotheses which is to align goals in a decision process where agents (both software and human) have preferences. Prior to exploring a new scheme for negotiation, a range of possible conflicts in a generic product development environment is explored.

2.6 Identification of research issues

In the context of collaborative product development the survey identified two parallel approaches based on CSCW and software agents. In CSCW based approach, conflict resolution is a task left for humans to resolve. This, as highlighted in the review, may not always lead to rational solutions (eg., parametric design problems involving a large number of variables) since this is limited by human cognitive skills. Based on a critical review of research in agent based design and manufacturing, conflict resolution between software agents emerged as a key issue which needs further investigation. To address this issue further, additional review was carried out on the following two inter-related strands:

- Synthesis of software agents;
- Schemes for resolving conflicts between agents.

It is evident from the review that very little research has been conducted which directly addresses the need for a rational process of conflict resolution in a collaborative product development process where agents have individual preferences [as highlighted in Figure 2.4].

It is inevitable that agents have individual preferences towards a common goal in any product development process. This often leads to conflicts between the agents and will need to be resolved in a timely manner. Though the software agent research has highlighted conflict resolution as a key issue, there is very little evidence of formal agent based schemes for conflict resolution applied in the context of a case study. The existence of conflict resolution schemes (such as those based on game theory and constraint relaxation) in the literature outside software agent domain is, however, acknowledged. There is a need for demonstrating the usefulness of rational conflict resolution schemes that could be applied to conflicts involving both software agents and human decision makers in the context of collaborative product development. This is aimed towards enhancing the nature of group decision making by improving rationality and maintaining preferences of agents.

This leads to the principal objective of this research: to develop a scheme for conflict resolution between agents in a collaborative process. To address this objective, two inter-related hypotheses of this research were formed as shown in Figure 2.6.

The main issue in the present research is the development of a scheme for resolving conflicts in an agent-supported product development process. This, in the author's view, is aimed at improving upon some of the methodological limitations suffered by previous research to conflict resolution in collaborative product development. The two hypotheses form the basis for developing the work in the next two Chapters. The hypotheses are tested through case studies in Chapters 5 & 6.

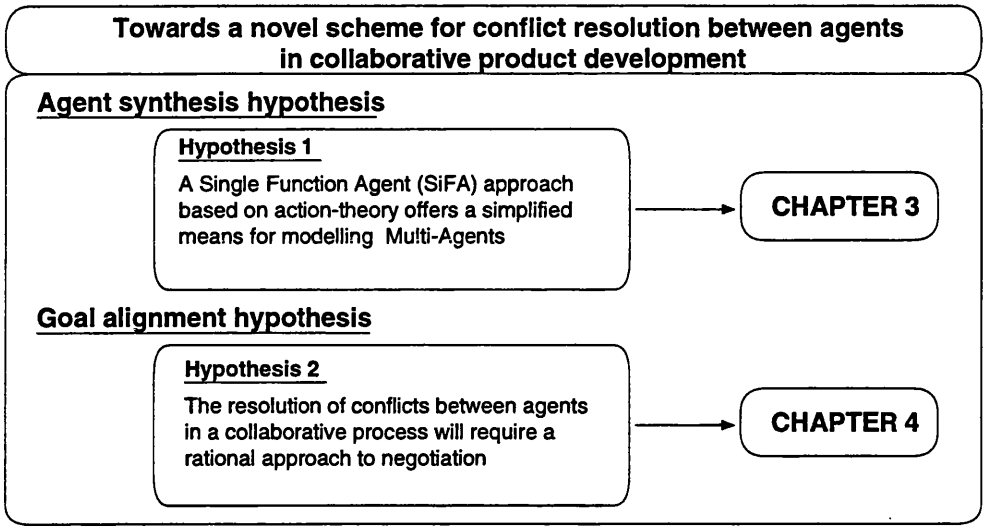


Figure 2.6: Hypotheses of this research

2.7 Summary

A review of research in collaborative product development based on software agents in the context of conflict resolution has been presented. In particular, this review covered two distinct areas: synthesis of software agents and conflict resolution schemes. This led to the identification of outstanding research issues and the formation of two hypotheses of this research. The following chapters are aimed at building new models for agent synthesis and conflict resolution based on these hypotheses and testing them through engineering case studies.

Chapter 3

Modelling of software agents

3.1 Introduction

This chapter is aimed at the modelling of software agents. A review of the SiFA approach by Berker and Brown (1996) presented in Chapter 2 ignored important issues such as the transition from SiFAs to Multi-agents (MAs). In this chapter the spotlight will be on the following key issues:

- An investigation into the Single Function Agent (SiFA) approach by Berker and Brown (1996) for modelling software agents;
- Modelling of software agents using the theory of communicative action;
- How SiFAs combine to form Multi-agents.

Emphasis is placed on examining how Multi-agents (MAs) are formed as a combination of Single Function Agents (SiFAs). This is important to gain a clearer understanding of the agents' actions. The internal structure of a software agent is explored in the context of an example of a material agent. The chapter concludes

by highlighting the key differences between the present and key previous research relating to the approach for modelling software agents.

3.2 SiFA model by Berker and Brown

3.2.1 Model description

SiFAs are specific for a task and information associated with them is specialised. Each SiFA can be considered as a building block of a Multi- agent system. Though Berker and Brown (1996) presented knowledge-based approach to model SiFAs, this was not systematically evaluated or applied to a design problem. They presented seven types of SiFAs associated with each design attribute as shown in Figure 3.1.

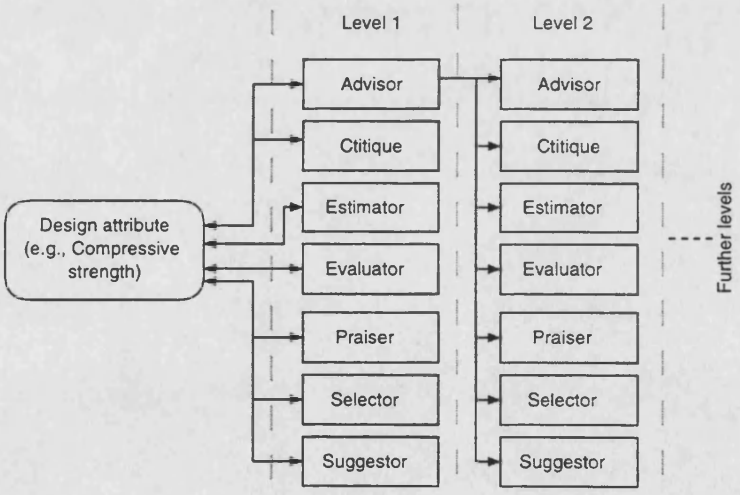


Figure 3.1: Single Function Agent (SiFA) model by Berker and Brown (1996).

The functions of seven types of SiFAs shown in Figure 3.1 are summarised as follows:

- **Advisor:** An advisor agent produces the values of a design variable using well defined parametric relations. For example, in a specific problem, an

advisor agent would generate the values of shear stress.

- **Critique:** A critique agent performs a safety check on the parametric values such as maximum allowed stress level. For example, for a given material and loading conditions on a structural member, static deflection could be a criterion for criticism. If the critique agent observes any violation of such criteria, the agent issues a warning to other agents.

- **Estimator:** The estimator agent estimates the outcome of a problem with insufficient description. This estimation of a design attribute is approximate and usually given as a range (e.g., varying between 10-15%).

- **Evaluator:** Evaluates the goodness of a variable and is usually expressed as a percentage of the ideal value.

- **Praiser:** In a scenario where there are several constraints to be met and where the design values are successful in satisfying only some of these constraints, the praiser agent will comment only on the satisfied constraints. The remaining unsatisfied constraints are object for the Critique SiFA.

- **Selector:** A selector agent is responsible for selecting parametric information from a list of possible values. For example, for a specified material, a selector agent would return the values of material properties such as modulus of elasticity or Poisson's ratio.

- **Suggestor:** The suggestor agent resolves conflicts that may arise between any two types of SiFAs.

Berker and Brown (1996) proposed a hierarchical model of SiFA's such that for each type of SiFA presented in Figure 3.1, there can be one or more further sets of SiFAs at deeper levels of abstraction. At each of these levels appropriate knowledge needs to be present for execution of tasks at that level. For a given case, several of these levels of abstraction may not even exist and it is necessary to analyse these levels in the context of an engineering design problem.

3.2.2 Analysis of the model with example of spring design problem

Since Berker and Brown (1996) did not provide a detailed example of their model, the author has considered the analytical design of a spring [Figure 3.3] here to evaluate their SiFA model. In particular, the design case is aimed at exploring levels associated with a spring design variable which is shear stress (τ_s). Figure 3.3 shows the expanded levels associated with (τ_s) Advisor(1-1) alone. For the sake of clarity, only the Advisor SiFA is expanded to the first four levels.

Figure 3.2: Schematic of a helical spring

At level-1, Advisor(1-1) generates the value of τ_s based on available analytical relations. In this case, this parametric relation is defined by $\tau_s = \frac{8F_sD}{\pi d^3}K_s$; where shear stress ($\tau_s$) is a function of spring coil diameter (d), mean diameter (D), solid force (F_s) and spring constant (K_s). With respect to Advisor(1-1), agent behaviour is more deterministic, ie., the agent generates a quantitative value for τ_s. Since, at level-1, the outcome for τ_s is predictable, the functions of Selector(1-

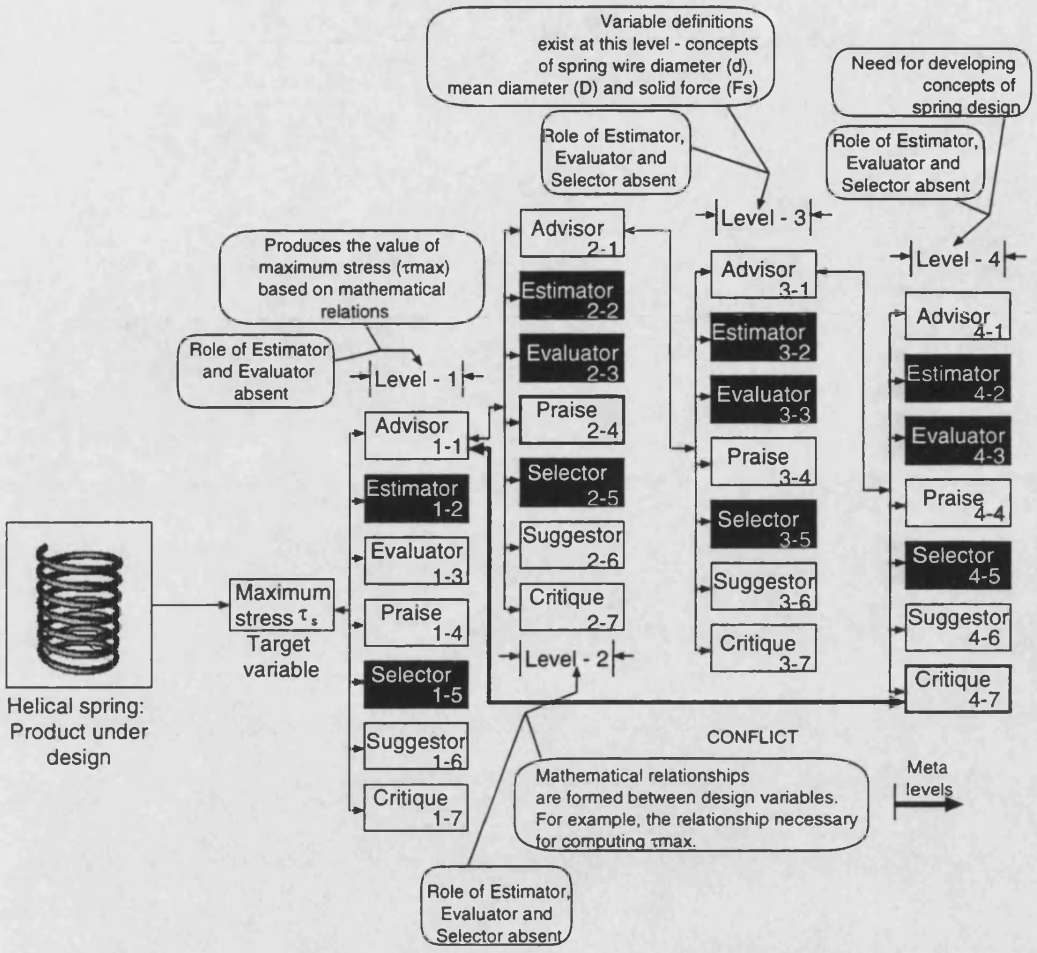


Figure 3.3: Decision levels in SiFA model for the spring design problem

5) and Estimator(1-2) are not required. Critique(1-7) could make use of IF-THEN rules to check a constraint such as $\tau_s \leq \tau_{max}$. If shear stress value exceeds the allowable stress (τ_{max}) value, a conflict is detected between Critique(1-7) and Advisor(1-1) SiFAs. Though Berker and Brown did not present a mechanism to resolve such conflicts, the author believes that Suggestor(1-6) could make use of protocols based on game theory.

Level-2 presents a slightly different case. Unlike the level-1 Advisor(1-1), Advisor(2-1) is concerned only with establishing mathematical relationships leading to the analytical expression for evaluating τ_s. As at level-1, the roles of Estimator(2-2), Evaluator(2-3) and Selector(2-5) have been diminished to zero. The nature of the outcome at this level is not quantitative. For example, the

Critique agent(2-7) may offer its criticism on Advisor(2-1) based on dimensional analysis (such as based on Buckingham's π theorem). It can be seen that the outcome at level-2 is qualitative; an aspect not acknowledged by Berker and Brown.

Level-3 as shown in Figure 3.3 presents even more generic situation. The role of Advisor(3-1) is seen as realising the parametric terms such as wire diameter (d), etc., in a context such as the design of a spring. The key objective here is to acquire knowledge leading to spring design. The roles of Estimator, Evaluator and Selector are absent at level-3. The role of Critique(3-7) agent is to analyse a design concept rather than a design attribute (e.g., allowable stress) as at level-1. Thus Critique(3-7) agent could offer its criticism on a design philosophy or the need for using springs in a given context. As can readily be seen, the Critique SiFA is no longer as finely granular as the Critique SiFA at level-1; level-3 SiFAs address much broader issues.

Level-4 presents a case of further meta-level abstraction. The role of Advisor(4-1) at this level is to identify the need for the design of a spring. The knowledge contained in each of these SiFAs at this level has not been explored by Berker and Brown.

Interestingly, Berker and Brown (1996) allow possibilities of conflicts across all levels as highlighted in Figure 3.3. However, this appears to be unrealistic due to the lack of global negotiation strategies. They localised the generic negotiation knowledge intertwined with design knowledge at that level. This makes the application of generic negotiation strategies difficult. It is worth noting that local strategies lead to sub-optimal solutions as rightly stated by Klein (1993)[1]. In this case even the *existence of* conflicts between Advisor(1-4) and Critique(4-7) is questionable due to the lack of a common goal between the agents highlighted in Figure 3.3. In the view of Berker and Brown, violation of design constraints is

[1]Klein(1993) suggested that conflict resolution knowledge should be independent of domain specific design knowledge in order to allow the generation of optimal solutions.

the way conflicts are defined and there is no consideration of conflicts due to differing preferences. The Suggestors at level four and level one could use strategies that may not lead to a unique settlement between conflicting agents. It is not even obvious what the contents of Suggestor(4-6) are since the conflict resolution knowledge at higher levels is not explained by Berker and Brown. The limitations in the Berker and Brown model are presented next.

3.2.3 Limitations of the model

To summarise this analysis, the following limitations are identified in Berker and Brown's SiFA model:

- Berker and Brown's SiFA approach does offer benefits of visualising an agent's internal structure, though in a highly pre-configured knowledge-based manner.
- The number of SiFAs associated in a design problem can be expressed as, $n7^{m_i}$, where n is the number of design variables at each level and $(m_i = 1....k)$ is the number of levels associated with each design variable; $k, n \in \Re$. The preceding analysis reveals that the presence of such unlimited levels could become unwieldy even for a simple problem involving only a few design variables. For $n = 10$ and $m_i = 4, \forall i$, the number of SiFAs would be 23060. In addition, Berker and Brown did not present a justification for the existence of such a large number of levels.
- As the number of levels increases, the information contained in each SiFA tends to become more generic thereby losing its finely granular nature.
- Conflict resolution strategies adopted by Berker and Brown seem to be implicit at each level and they tend to be specific for each design case. There is a need for conflict resolution strategies that are independent from the domain.

- Berker and Brown treated conflicts as violations of design constraints; an assumption that is unwise since real world conflict situations are depicted as disagreements between two or more agents due to different preferences.

Some of these limitations are overcome in the new SiFA model presented in the next section.

3.3 A new model for Single Function Agents

The author's research attempts to address the limitations identified in Berker and Brown's model. Specifically, it will address the following:

- Reduction in the number of levels of SiFAs;
- Explicit representation of agents' actions;
- Transition to Multi-agents;
- Strategies for conflict resolution and reasoning in a domain.

This section presents an approach for modelling Multi-agents using a new model for Single Function Agents (SiFAs). Before presenting the proposed model for SiFA, Habermas's theory of communicative action is outlined, on which the new model is based.

3.3.1 Theory of communicative action

Habermas's theory of communicative action (Habermas, 1984) deals with categorising group interactions between humans in a collaborative problem solving

environment. The action theory as described by Habermas is a web of coordinated actions taken by humans in a collaborative problem solving environment. Such coordinated actions will include a common language for communication, shared ontologies, rules and constraints in a group working scenario. In this context, Habermas proposed a set of rules and resources which are necessary to support social activities between humans in an organisation. The four action categories of Habermas's theory are stated as follows:

- Instrumental
- Communicative
- Discursive
- Strategic

Instrumental action deals with manipulating and controlling objects using technical knowledge and tools in the environment. The technical knowledge and tools are known as allocative resources. In an organisational context these resources are inanimate objects such as software tools or manufacturing machines.

Communicative action occurs when participants of a group process get together to exchange knowledge in order to solve a task. This will require a medium of communication through which participants can share their ideas and understanding. The participants must possess technical knowledge of the problem and language. Knowledge of the language includes the syntax and semantic level of understanding.

Discursive action focusses on analysing the problem including the instrumental and communication categories that assist the discursive process. Therefore, discursive action is more specific in nature compared to communicative action. Since major decisions are made during a discursive process, use of precise ontologies and rules is critical.

Strategic actions relates to the highest level of problem solving in a group. Each action in a groupwork environment is based on some strategy the participants adopt towards achieving a goal. Participants have the knowledge of the rules and constraints involved in a strategic action. However, due to the existence of individual preferences among participants, conflicts are inevitable. Habermas defined certain rules that govern strategic action in a conflict situation. These rules provide participants with equal chances to express their views, to reason based on facts and to show no bias towards a particular participant.

Habermas's theory has been used to analyse many issues related to organisational workflow. For example, Ngwenyama and Lyytinen (1997) used this technique to study social action in a Computer Supported Cooperative Work (CSCW) based group work environment. They measured various social action situations (e.g., group decision making) met within organisations. In what follows, Habermas's theory is used to categorise actions in a SiFA which will be then mapped to Multi-Agents.

3.3.2 Action theory based new SiFA model

Habermas's theory of communicative action is applied here to software objects and has been used to model SiFAs for design. In particular, each action category is explored to identify the knowledge necessary to support SiFAs' behaviour. The proposed approach to SiFA model is novel in two respects:

- The membership of the set of SiFAs (what can be a SiFA?);
- The composition of a SiFA (what are its capacities?).

It consists of the following steps:

- Specification of a design;

- Formation of a graph of design activities;
- Mapping of each activity to a specialist agent;
- Identification of design variables controlled by each specialist agent;
- Mapping variables to SiFAs;
- Associating actions to each SiFA.

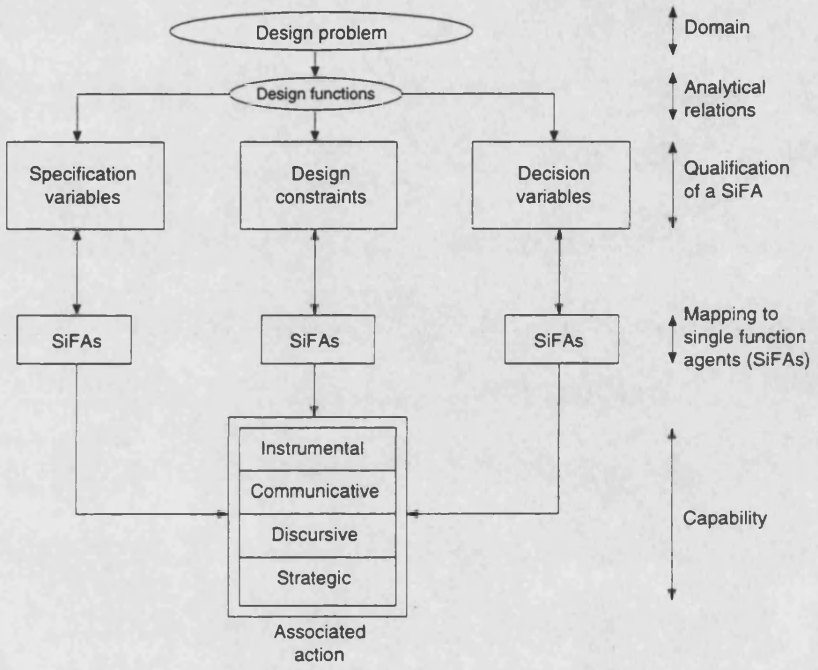


Figure 3.4: New action theory based SiFA model

As shown in Figure 3.4, a design problem is specified as a set of relations which are expressed as functions of design variables. These functions include mathematical equations, qualitative constraints and heuristic rules. Once the problem is specified, certain variables within these functions are mapped on to individual SiFAs. Only specification variables, decision variables and constants are mapped to SiFAs[2]. Once the individual SiFAs have been identified, their capabilities are defined by associating one or more action categories with each SiFA. The choice of action categories associated with a SiFA depend on its type:

[2]In the SiFA model by Berker and Brown (1996), all design variables (including constants, specification and derived) were mapped to SiFAs. Unlike the Berker and Brown model, the new model does not treat all variables as SiFAs.

- Constants: Instrumental action, Communicative action
- Specification variables: Instrumental action, Communicative action
- Decision variables: Discursive, Strategic, Instrumental and Communicative action.

The instrumental action associated with a constant or variable would involve storage and retrieval of its value before it can be communicated to other agents. In some cases the instrumental action may not be visible to an observer. In a more explicit case the instrumental action will involve a designer specifying an initial value or making some parametric choice.

Communicative action involves exchange of values (of a constant or variable) with other SiFAs. It also involves the preparing of the communication object to be exchanged (query/reply and their format).

The discursive action deals with explicit rules or procedures to support the evaluation of a decision variable. This is the action which would be responsible for most of the computational effort of a SiFA.

The strategic action category deals with checking the violation of design constraints. To support this type of action certain IF-THEN rules are specified. The new SiFA model is next explored in the context of the spring design example previously considered in Section 3.2.2, to allow its comparison with Berker and Brown (1996).

3.3.3 Analysis of the new SiFA model using the spring design problem

In this section the new SiFA model is illustrated using the spring design problem. Each action category of a SiFA is explored to identify the knowledge necessary to support the SiFA's behaviour. Recalling equation for shear stress τ_s in spring design [Section 3.3.2], τ_s is the spring design function under consideration. There is a set of five design variables and constants associated with τ_s based on the analytical relationship. A SiFA is associated with each of these as shown in Figure 3.5. Table 3.1 presents the different action categories associated with these SiFAs.

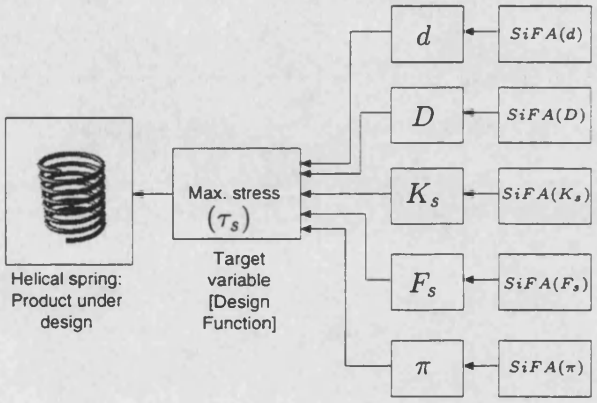


Figure 3.5: New SiFA model applied to attributes in spring design

Table 3.1: Categories of action in the spring design example

Variables mapped to SiFAS	Categories of action associated with the evaluation of τ_s			
	Instrumental	Communicative	Discursive	Strategic
d	Threaded methods: - retrieval - display	Shared class: - accessing the value - use of synchronised methods	Specific methods: - methods for simple computation (e.g., percentage change) - getText() method in Java for retrieval - return methods for static variables	$d_{in} \leq d \leq d_{out}$
D				$D_{in} \leq D \leq D_{out}$
K_s				$K_s^{in} \leq K_s \leq K_s^{out}$
F_s				-
π			- Math.PI in Java - return methods if declared as a static variable	-

Implementation of the new SiFA model

In software terms, the instrumental action would involve the control and manipulation of software objects. This instrumental action can be explicit or implicit depending on the nature of the object. For example, in the case of a human designer interacting with a software via a GUI, the instrumental action could involve the designer clicking buttons in the GUI or specifying the preferences for d or F_s; the agent responds to this in a pre-determined manner by opening up a dialog box. Since SiFAs operate at a very much finely granulated level, the instrumental actions are not visible to an observer. As Table 3.1 indicates the instrumental action would involve the use of threaded control mechanisms for the exchange of data (e.g., a read-write scenario). The result of such an instrumental action (values for d and D) could be made visible via, say a GUI.

Agents may exchange information by communicating synchronously or asynchronously. The advantage of synchronous communication is that it helps the agents to communicate interactively. Data to be shared between SiFAs is encapsulated and maintained using synchronised methods of a shared class as shown in Figure 3.6.

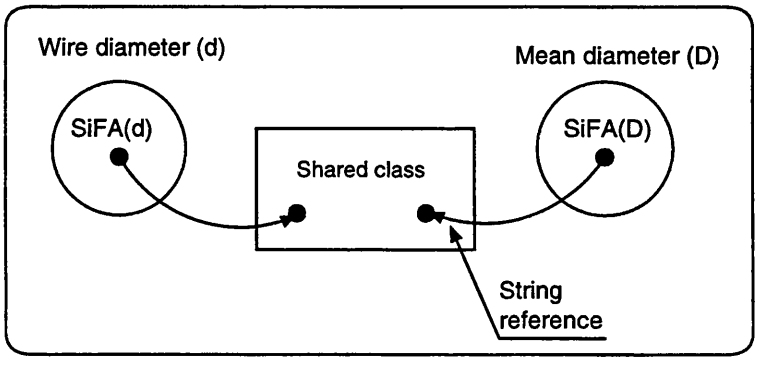


Figure 3.6: A shared class mechanism

The shared class mechanism acts as a blackboard through which SiFAs communicate. One important aspect during the process of communication is that the ontology is also shared between SiFAs. For example, the evaluation of vari-

able τ_s will need to access the item with appropriate string references such as `wire_diameter` and `mean_diameter`.

The SiFAs resort to discursive action to execute a task. Each SiFA has built-in methods to support its task solving. Since all the variables associated with τ_s are either constants or specified for a given problem, the methods associated with the SiFAs are relatively simple. In the present case as indicated in Table 3.1, these methods could be simple retrieval functions (e.g., Math.PI in Java).

Strategic actions in SiFAs are more specific in nature. At SiFA level, this category of action deals with checking violations of design constraints, if any. The strategies allow the SiFAs to probe the results of discursive action. These strategies are usually `IF-THEN` rules. In Table 3.1, for the variable K_s an inner and outer limits K_s^{in}, K_s^{out} are set to support strategic action. Similarly, the value of d could be valid only in the range $d_{in} \leq d \leq d_{out}$ for a given specification. Therefore the strategic action of SiFA(d) is meant to ensure that the value of d lies in the safe range. The exact values for this range is determined by the physical nature of the problem. Table 3.1 shows that all SiFAs do not need a strategic category. For SiFA(F_s), there is no need for any strategic category as the value of the solid force F_s is specified by the designer.

Having presented details of the new SiFA model, its advantages over the model of Berker and Brown are presented next.

3.3.4 Advantages over Berker and Brown SiFA model

The action theory based new model for SiFAs offers advantages over the approach of Berker and Brown (1996). The following points highlight the key advantages:

- Reduced number of levels;

- Reduced number of agents;
- Explicit representation of agent's actions.

Reduction in the levels associated with a SiFA leads to a simpler form of representation of the agent model. In the present approach there is only one level of decision compared to that of Berker and Brown (1996). Therefore it is eaiser to support the action categories within that one level. For example, referring to spring mean diameter (D), considered in Section 3.3.3, it is easy to derive methods necessary to evaluate D. Similarly, F_s is initially specified by the designer and will require only methods to read from the user input. The reasoning aspect in the agents behaviour was kept limited to the constraint violation as a practical design consideration rather than meta-level abstract reasoning in Berker and Brown model. Hence the procedure for the identification and implementaion of methods necessary to support each SiFA is simplified compared to Berker and Brown's approach.

Due to the presence of reduced number of decision levels, the number of SiFAs necessary is reduced. This is due to the elimination of unwanted meta-level abstract reasoning. Thus the methods to support agent's actions are deterministic. Explicit representation of actions associated with each agent assits in organising the actual methods necessary to support such actions. The author's approach presented four action categories each fulfilling a unique function, whereas Berker and Brown (1996) did not analyse actions associated with their agents. Such an analysis is essential in implementing the software agents systematically. The system designer can thus identify the necessary methods and their modularization (e.g., synchronised methods in a communication scenario and domain specific computational methods). As Table 3.1 highlighted, there was no need for strategies associated with certain SiFAs [SiFA(F_s) and SiFA(π) in Table 3.1]. In contrast, the approach of Berker and Brown will involve unlimited levels of abstract reasoning with the additional difficulty of identifying the necessary methods to support these levels.

The other aspects not covered by Berker and Brown (1996) which are considered here are the following:

- Strategies for conflict resolution;
- Transition to multi-agents.

Though Berker and Brown claimed their SiFA model to be in the context of conflict resolution, they did not present a scheme for conflict resolution. Their ideas for conflict resolution were largely informal with a cursory mention in the context of wine glass design. The author's research closely relates the agent synthesis with conflict resolution as a key issue which is the topic of Chapter 4.

Single Function Agents (SiFAs) operate at a finely granular (design variable) level and their application in the context of problem solving is realised at a higher level (e.g., multi-agents). This is an important aspect in modelling agents which Berker and Brown (1996) ignored to address[3]. The author's research has addressed the transition from SiFAs to multi-agents which is the object of next section.

3.4 Transition from new SiFAs to Multi-Agents

Several Single Function Agents (SiFAs) put together form a Multi-agent. But considering the finely granular nature of SiFAs and coarse grained Multi-agents, it is necessary to identify how SiFAs at the parametric level combine to form a collection of Multi-Agents (as seen in many Multi-Agent systems). With a view to gaining such an understanding, this section describes the formation of

[3]Berker and Brown (1996) assumed all variables in their design problem as SiFAs. In the new SiFA model presented here, such a view is not appreciated since the complexity in determining variables in a design problem are not the same. In a small-scale problem such as the wine glass design considered by Berker and Brown (1996) which involves only a few variables (less than 5), this distinction may not be obvious but for a large problem involving several variables, their approach can become unweildy.

Multi-Agents. In this discussion, the term SiFA refers to the new model of SiFAs proposed in Section 3.4.

3.4.1 How SiFAs combine to form Multi-Agents?

Variables form the base level description of a design problem. It is by using these variables that explicit engineering relationships are often built. In a typical product development phase such as detailed design, explicit identification of a single variable at the base level is often not obvious. It is often necessary to identify a *set of variables* embedded in design functions (such as design cost, strength, etc.) to describe a desired phase. Classification and group identification of design variables are often used to organise various design activities which assist in backtracking [Kusiak and Wang (1995)]. An extended approach is adopted here where the transition from SiFAs to Multi-agents is carried out in successive stages with increasing degree of complexity. These stages are summarised below:

- Single Function Agents (SiFAs): apply to variables (including decision variables) that are specified at the start of a design process. SiFAs also refer to engineering constants;
- Simple Function Agents (SimFAs): they represent simple design functions that are formed by a mathematical or fuzzy relationship between design variables;
- Multi-Function Agents (μFAs): the collection of several mutually-related simple design functions leads to μFAs. Each μFA represents a design perspective with a design preference which characterises its autonomy. The design preference is usually captured in the form of utility functions. For example, for the spring design agent a preference could be to minimise stress;

- Multi-Agents (MAs): They operate at a much higher level of abstraction. they consist of several μFAs. For example, a Multi-Agent for design of an product would include several μFAs for individual parts or sub-assemblies.

Figure 3.7 shows the transition of SiFAs to Multi-Agents with systematic progressions through the intermediate stages. Each design variable is governed by a SiFA at the base level. A related group of SiFAs forms a simple function which leads to a Simple Function Agent (SimFA). For example, shear stress(τ_s) in the spring design problem is represented by a simple function agent SimFA(τ_s). A simple function agent is formed as a result of tight coupling between design variables which are SiFAs.

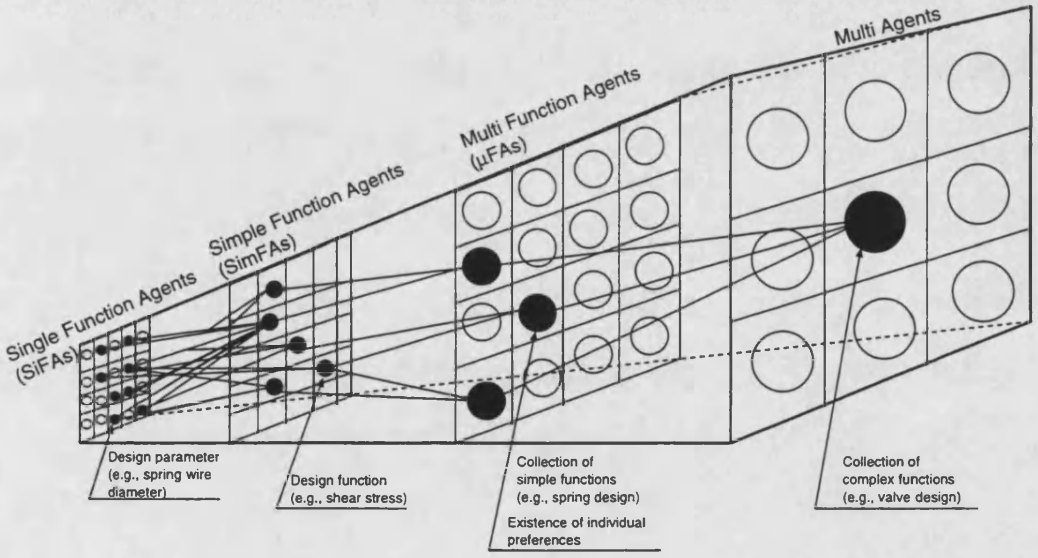


Figure 3.7: Stages of transition from SiFAs to Multi-Agents

For example, in spring design there would exist several SimFAs for computing stress, spring factor, solid length, etc. Several such Simple Functions Agents (SimFAs) group together to form what is defined as a Multi Function Agent (μFA), for example, an agent for spring design. A collection of such μFAs lead to Multi Agents (MAs). For instance, a relief valve design would involve the design of springs, valve body, etc. Thus MAs are usually complex (i.e. wide range of capabilities) and are composed of several specific and generic μFAs.

3.4.2 Structure of a Multi-Function Agent

A Multi-Function Agent(μFA) is composed of some generic components for task processing (e.g., communication and negotiation protocols) and some specialised components to achieve domain-specific objectives.

Generic components: The μFAs communicate via the exchange of action-oriented KQML messages [Finin, *et al.*, (1995)]. Some of the performatives that imply agent's action are *tell*, *ask* and *disagree.* Apart from KQML support for communication, the Multi-Function Agents also provide a user interface for interaction with human agents.

The μFAs offer reasoning (based on qualitative and quantitative dependencies) support to parametric design agents. In addition, negotiation support is also provided to resolve conflicts on a common goal between μFAs. A detailed discussion on reasoning and negotiation mechanisms is the subject of Chapter 4.

Specific components: Design agents use methods that are specific to a problem domain. For instance, in the case of a spring design problem, the designer agent would use special functions such as shear stress, Wahl factor, etc. Each designer agent would also have a set of problem specific variable s and constraints. The designer agent communicates with human agents via a GUI. The goals and constraints of each design agent are specified at the start of the design process.

3.4.3 An Example of a Multi-Function Agent

A material agent which was developed for this research is shown in Figure 3.8. The material agent is responsible for returning the properties of a specified material (e.g., AISI 1020 steel). The material agent is a μFA which has specific as well as generic components. The SiFAs and SimFAs associated with the material

agent are shown in Figure 3.9. The SiFA-based variables shown in the figure represent the base description in any material property set and they are generic to any material. The material agent has two key functions: retrieval of the material property set [`Material(material_name)`] and communication [`Tell, Ask`].

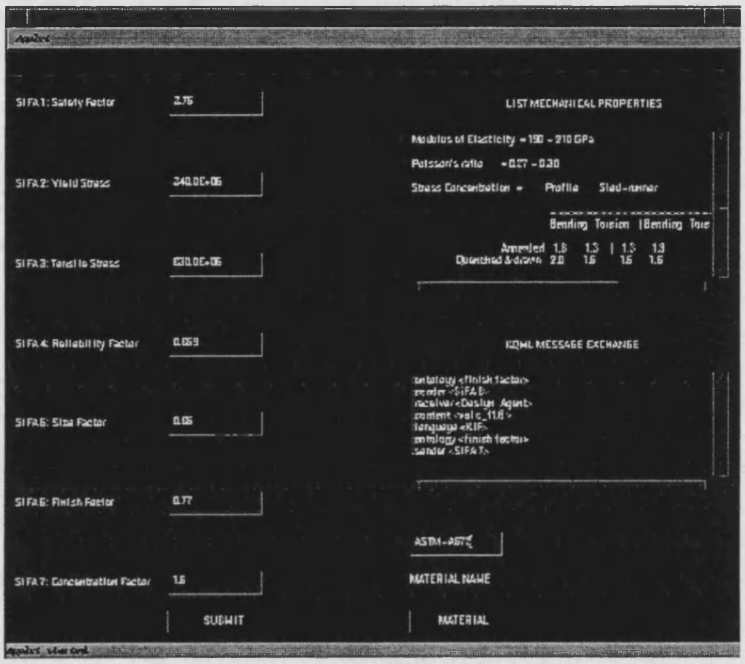


Figure 3.8: Material agent

The semi-autonomous material agent provides a simple GUI for interaction with humans. As shown in Figure 3.8, the GUI presents an area for displaying the retrieved material properties as the human designer clicks the appropriate buttons in the GUI (instrumental action). The communication between material and another Multi-Function Agent is captured in a message exchange area within the GUI. As the threaded communication takes place, the results are displayed in this area. The current implementation (to demonstrate the communicative action) uses *tell* and *ask* performatives.

An instance of communication between design and material agents is shown in Figure 3.10(a). The design agent uses the performative *tell* to query the material agent regarding the value of Poisson's ratio. The material agent uses the *tell* performative to reply with the value of Poisson's ratio as 0.3. Another instance

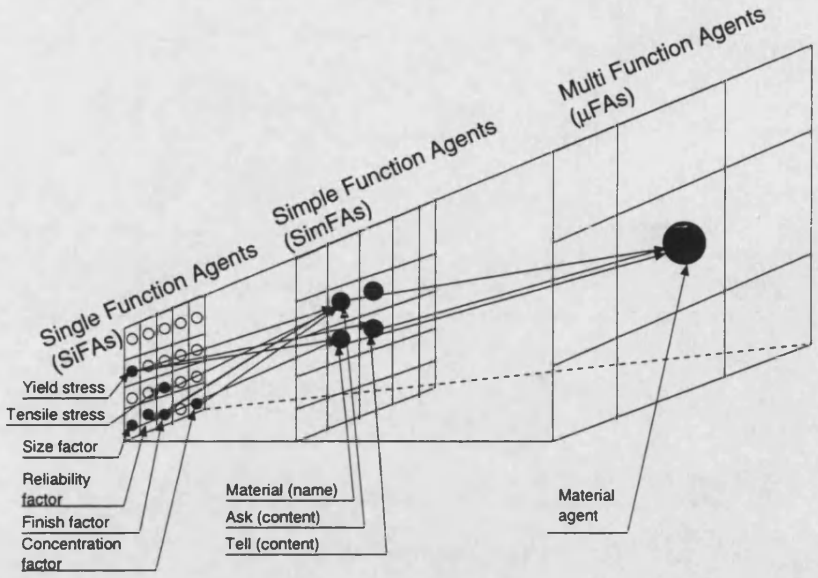


Figure 3.9: Transition from SiFAs to a μFA in material agent

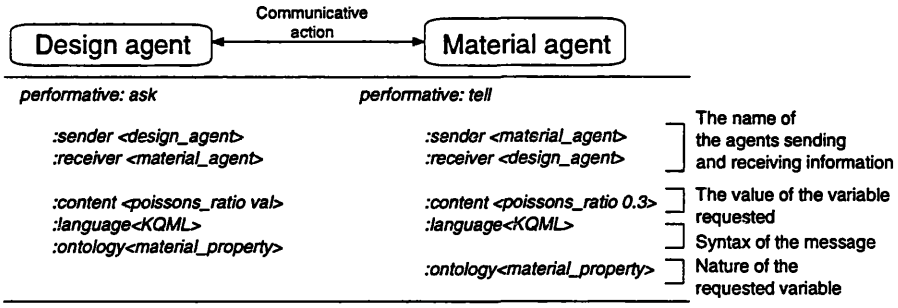
of communicative action is shown in Figure 3.10(b) in the context of a design agent sending *evaluate* performative to query the analysis agent about the state of stress in a structural member. Whereas Figure 3.10(a) uses simple message structure, Figure 3.10(b) presents a semantically precise message exchange. The figure shows that both kinds of communication are possible.

The material agent uses simple IF-THEN rules to check if a material property set is available and issues warning messages. Then the designer takes appropriate action and provides an alternative material specification. The example presented here shows how the transition from SiFAs to a μFA can be achieved in practice. The material agent will be presented in Chapters 5 & 6 in a broader context with other similar μFAs in a collaborative process.

3.4.4 An example of a Multi-Agent

To demonstrate the transition of SiFAs to Multi-agents, a Geneva mechanism design problem is considered. The details of this design problem are given in Appendix B. The design specification variables and some constants in Geneva

Example 1: The Design Agent sends "ask" performative to Material Agent to request the value for Poisson's ratio. The material agent uses "tell" to convey the requested value for Poisson's ratio.



Example 2: The Design Agent sends a request to Analysis Agent to query the stress due to torque loads on a structural member. The Design Agent uses the performative "evaluate" and the Analysis Agent replies with the performative "safe".

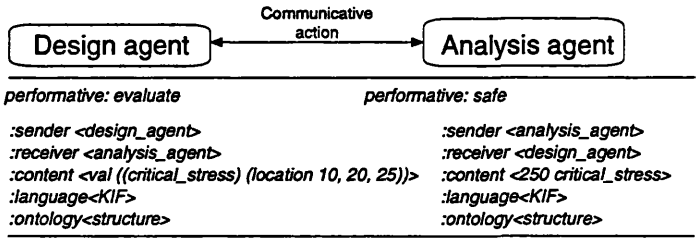


Figure 3.10: An instance of KQML message exchange

mechanism design are associated with SiFAs. For example, the specified design variables N_s, I_L, ω and certain material constants such as Young's modulus are associated with SiFAs. Such SiFAs combine to form a simple function agent (SimFA) as shown in Figure 3.11.

A simple function agent for wheel angle SimFA(β_0) is formed as a result of the combination of number of slots (N_s) and the mathematical constant π [Figure 3.11(a)]. Similarly for the driver angle α_0 as shown in figure 3.11(b)], a SimFA(α_0) is formed as a combination of number of slots (N_s) and π. Figure 3.11(c) shows a simple function agent for wheel slot distance (S) formed as a result of a combination of SimFA(C) and SimFA(β_0). A simple function agent for root stress (σ_R) as shown in Figure 3.12 is formed as a combination of several SiFAs and SimFAs.

Several such SiFAs and SimFAs combine to form Multi-Function Agents (μFAs).

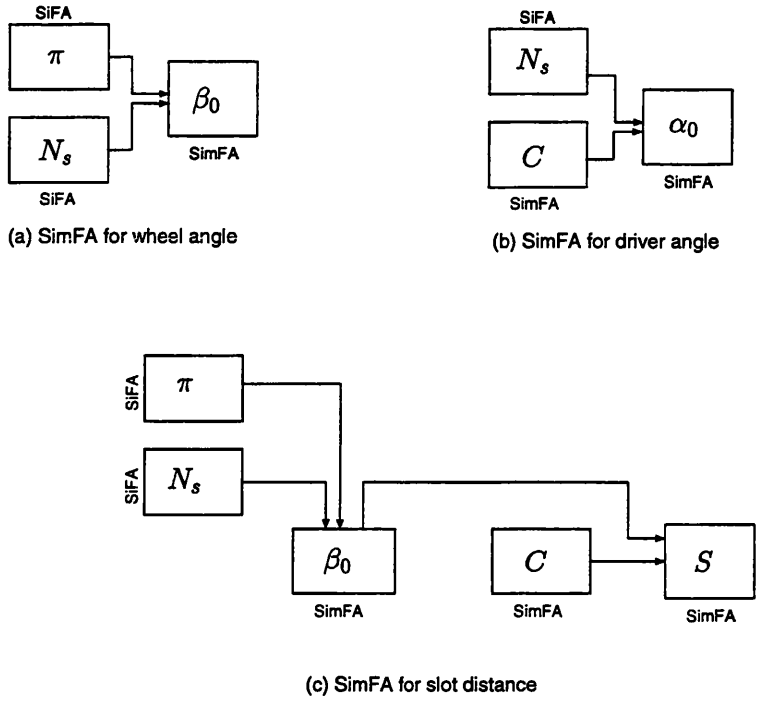


Figure 3.11: Formation of a SimFAs in Geneva mechanism design

In the wheel design case, SimFA(σ_R), SimFA(w_T), SimFA(S_w), etc. combine to from the wheel design agent which is a μFA. It is the support of these several functions that makes the wheel agent a multi-function agent. The transition from SimFAs to μFAs for the wheel and driver design is shown in Figures 3.13 & 3.14.

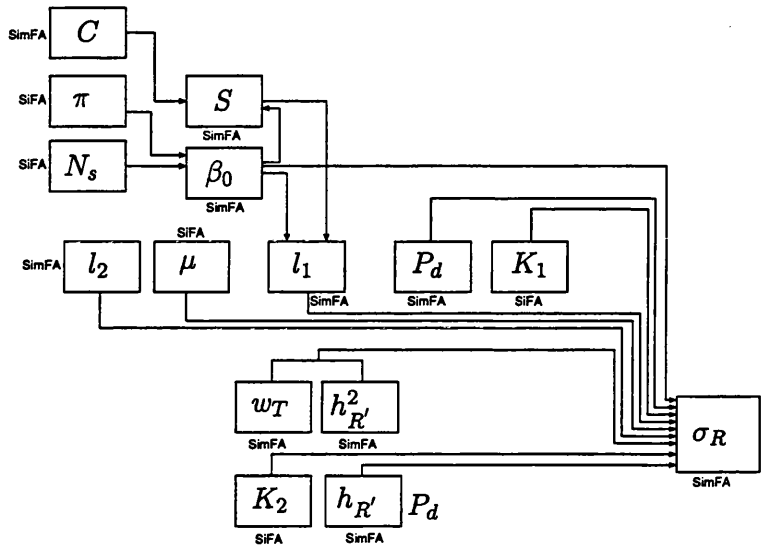


Figure 3.12: Formation of a SimFA(σ_R) in Geneva mechanism design

The wheel and driver agents (μFAs) combine to form the multi agent for Geneva mechanism design as shown in Figure 3.15. The stages of transition of SiFAs to a Multi-agent in the case of Geneva mechanism design are captured in Figure 3.16. The combination of such multi-function agents leads to a Multi-agent which is the design agent for the Geneva mechanism. The key SiFAs, SimFAs and μFAs in the Geneva mechanism design are shown in Figures 3.13 and 3.14.

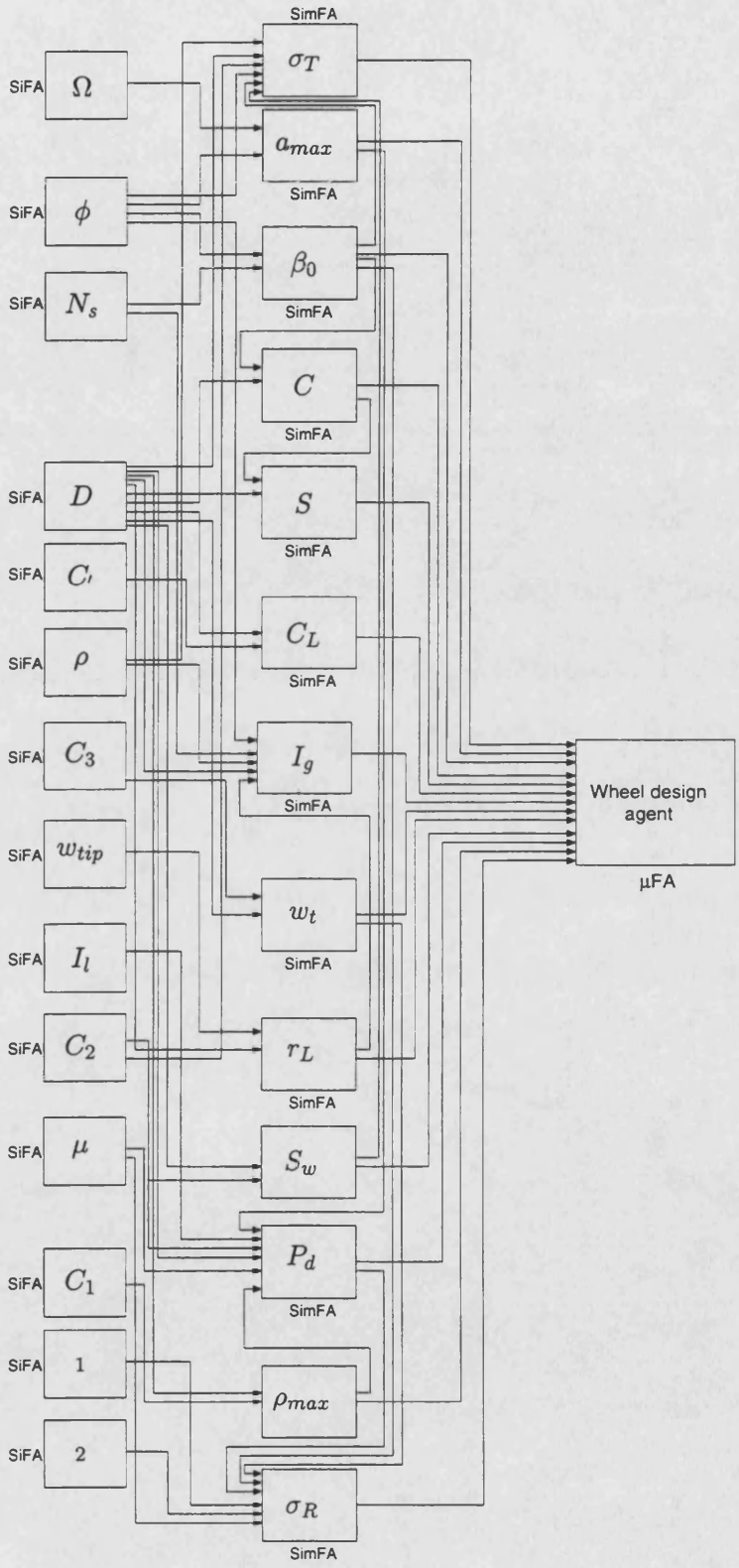


Figure 3.13: Formation of a μFA for wheel in Geneva mechanism design

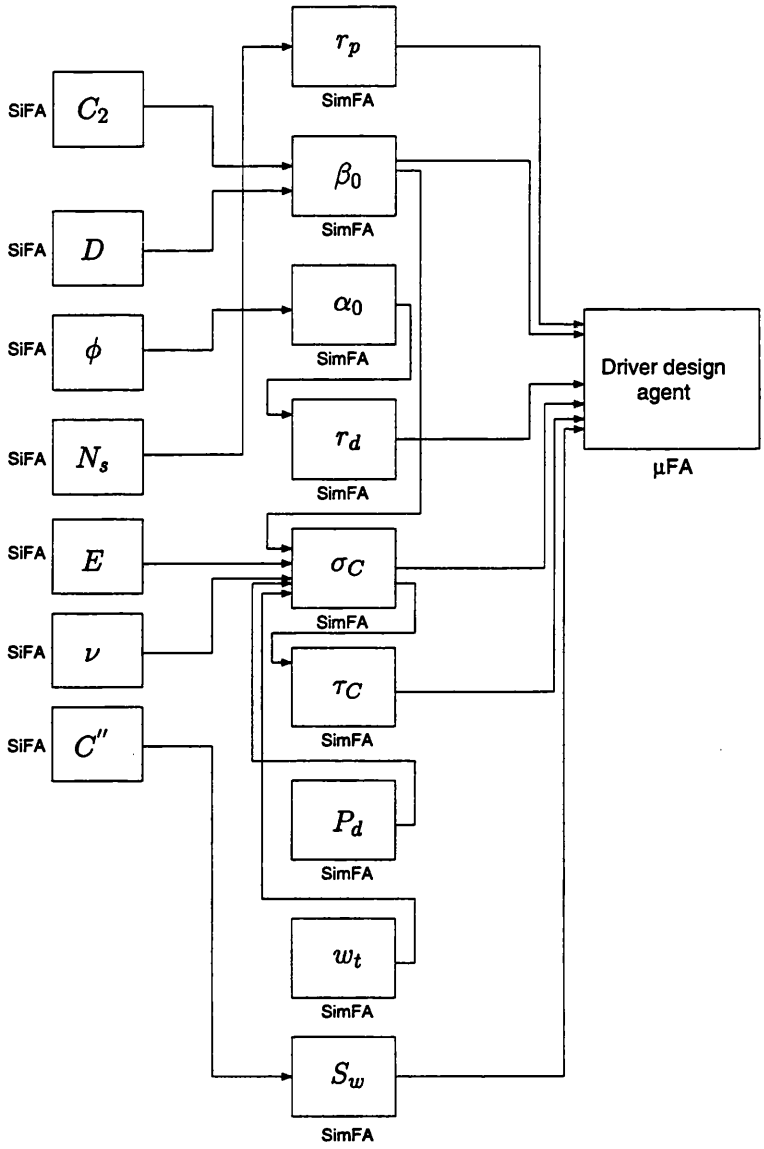


Figure 3.14: Formation of a μFA for driver in Geneva mechanism design

The GUI of the Geneva mechanism design agent is shown in Figure B.1. The agent is composed of generic as well as specific components similar to the structure of explained in Section 3.5.2. The Geneva mechanism agent will be presented in Chapter 6 in a wider context of a collaborative process.

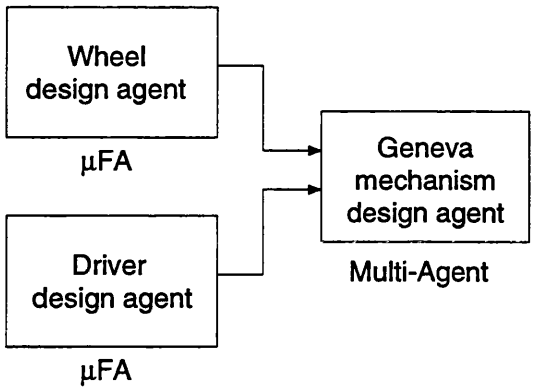


Figure 3.15: Formation of a Multi-agent for Geneva mechanism design

3.5 Differences from previous work

Key differences regarding SiFAs between the author's approach and that of Berker and Brown (1996) are drawn in this section. It highlights how some of the theoretical and implementational shortcomings in previous research have been overcome by the new model.

3.5.1 Theoretical differences

The distinction from Berker and Brown's SiFA approach is examined with respect to the following key characteristics:

- Number, levels and types;
- Scalability of SiFAs to form Multi-agents;
- Negotiation strategies;
- Communication;

Berker and Brown (1996) approach associates a hierarchy of SiFAs with each design variable, regardless of its primary or secondary roles. That amounts to

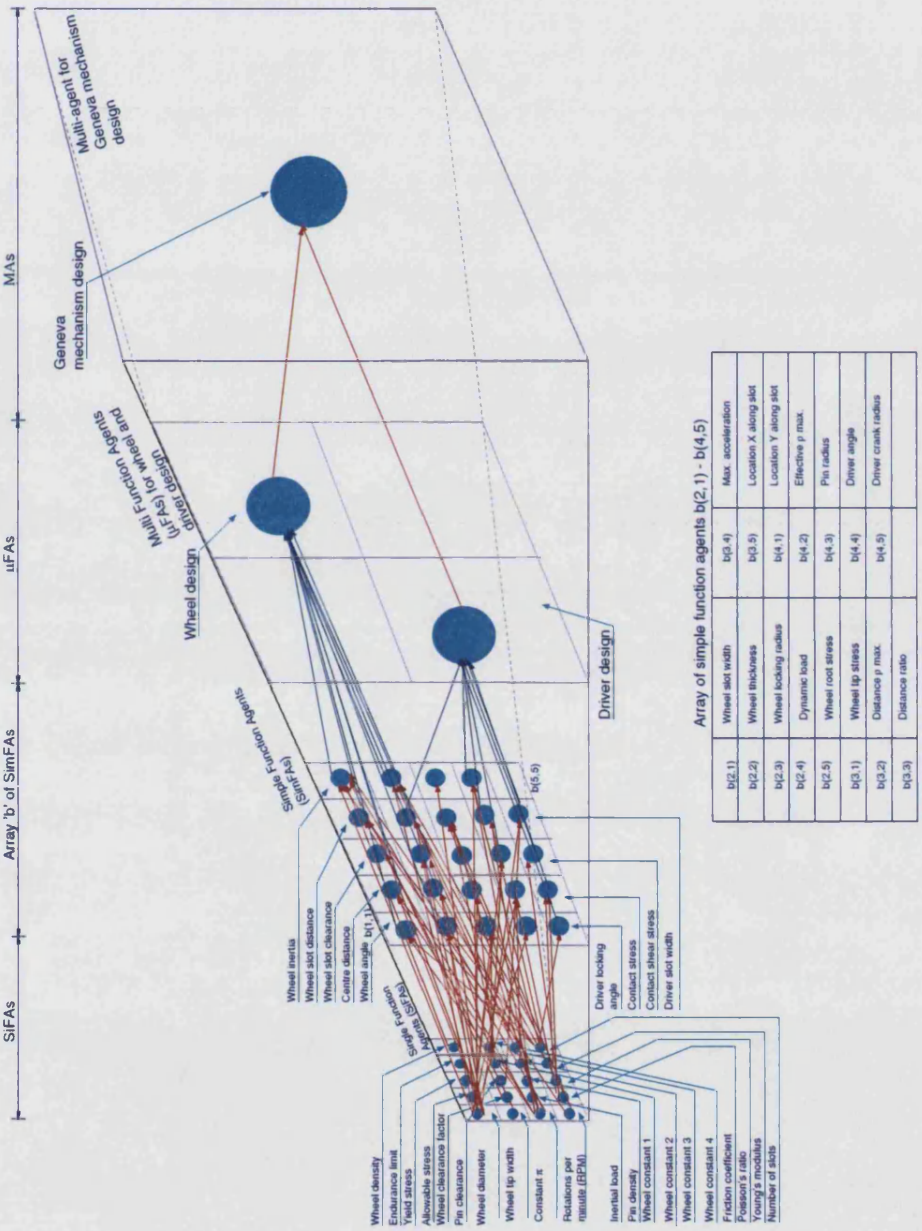


Figure 3.16: Transition from SiFAs to a Multi-agent in Geneva mechanism design

a very large number of SiFAs even in a simple design problem such as spring design. This research only creates a SiFA for the primary design variables and constants. Berker and Brown considered seven types of SiFAs (knowledge-based) associated with each design variable whereas this research considers only four types (action-based). As compared to the unlimited number of decision levels in the approach by Berker and Brown, the author's research requires only one decision level to ensure design integrity at the SiFA level.

Scalability of SiFAs to Multi-agents (MAs) was not addressed by Berker and Brown. The present approach analysed [Section 4.4] how SiFAs can combine to form Multi-agents and has been systematically realised through an example of the Geneva mechanism design agent developed for this study.

Though Berker and Brown considered negotiation between SiFAs, the strategies adopted for conflict resolution were very much problem dependent and hence lacked generality. The present research treats conflict negotiation as a generic issue. Further details are given in Chapter 4.

The communication mechanism presented by Berker and Brown is based on KQML. Each SiFA has its own independent knowledge regarding the syntax and semantics of the language. In the present approach, however, SiFAs communicate by means of a shared class where design variables representing common ontology are exchanged. At the Multi-agent level, the agents use KQML to communicate with other agents.

The key theoretical characteristics presented here influence the implementation of agents and are discussed next.

3.5.2 Implementational differences

Berker and Brown's SiFA approach presented unlimited decision levels associated with each SiFA. From an implementation point of view, this approach can become quite unwieldy even for problems with low to medium complexity. For example, communication, decision and task solving knowledge needs to be encapsulated in each SiFA.

The author's approach takes a different view as so much autonomy is not given to SiFAs at the parametric level. More importantly, the concept of unlimited decision levels is eliminated while preserving the design constraints. All the necessary support for task processing is encapsulated in each category of action in a SiFA. The Multi-agents modelled using such SiFAs still maintain their reference to categories of action. This helps in organising each agent's actions in a given situation whether at SiFA or μFA level.

3.6 Summary

A SiFA based approach for modelling complex software agents has been presented. A detailed analysis led to the identification of several drawbacks in the SiFA model by Berker and Brown(1996). The present approach addresses these drawbacks by means of modelling agents based on their actions. To this end, the Habermas's theory of communicative action has been applied to software objects and a new SiFA model has been proposed. A systematic transition from SiFAs to Multi-agents (MAs) has been demonstrated via an example of a design agent for the Geneva mechanism. Finally a brief overview of key differences from Berker and Brown's technique was presented. The proposed approach is elegant and simple. The next chapter will address the issue of conflict resolution between software agents.

Chapter 4

Conflict Resolution between Multi-Agents

4.1 Introduction

This chapter is aimed towards addressing the second hypothesis of this thesis which states, *"The resolution of conflicts between agents in a collaborative process will require a rational approach to negotiation"*. This chapter views collaborative product development as a negotiation process where the agents have individual preferences over a common objective that lead to conflicts. A review of previous research on conflict resolution in product development was presented in Chapter 2. The focus of this chapter is on developing a novel scheme for conflict resolution. The following key points are addressed here:

- Conflict resolution schemes based on game theory and dependency based reasoning.
- Development and implementation of a novel scheme for conflict resolution between agents.

4.2 Schemes for conflict negotiation

There is a variety of approaches available for negotiation depending on the type of conflicts as reviewed in Chapter 2. Here two approaches based on game theory and dependency based reasoning are considered in detail.

4.2.1 Game theory based schemes

Games are a composition of decision makers with set preferences on a collection of decision variables and strategies. Although this approach has been used mainly in the field of economics and strategic warfare, some of the work in the recent past shows its applicability in engineering design [Vincent (1981), Thomson and Lensberg (1989), Kanappan and Marshek (1993), Badrinath and Rao (1996), Rao, *et al.*, (1997), Hacker and Lewis (1998), Lewis and Mistree (1998)].

Four different game theory based methods, viz, Nash, Kalai-Somordinsky, Egalitarian and Utilitarian approaches are considered here. The Nash solution since its introduction in the 1950's has been very popular for the analysis of conflicts. The other solutions discussed here are some of the variations of the Nash solution [Lewis and Mistree (1998), Thomson and Lensberg (1989)]. As shown in Figures 4.1(a) 4.1(b) there are two agents (Agent_1 and Agent_2) with certain specified preferences described by their utility functions (U_1 and U_2). The possible joint course of action is determined by plotting each agent's utility functions as shown in Figure 4.1. The curve S in Figure 4.2 describes this joint course of action in a decision space which is assumed to be strictly convex.

- The Nash solution is defined by $N(S)$, such that $S \in \Re^n$; $N(S) = \max([\Pi_{i=1}^{2} U_i])$ is the maximum Nash product of agent utilities. $N(S)$ is also the undominated point on S such that it is the point of contact of the highest rectangular hyperbola $U_1 U_2 = k$ touching S and having

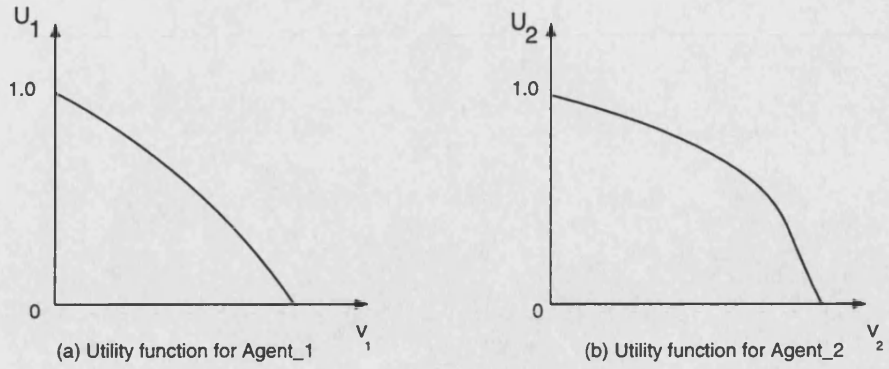


Figure 4.1: Utility functions for Agent_1 and Agent_2

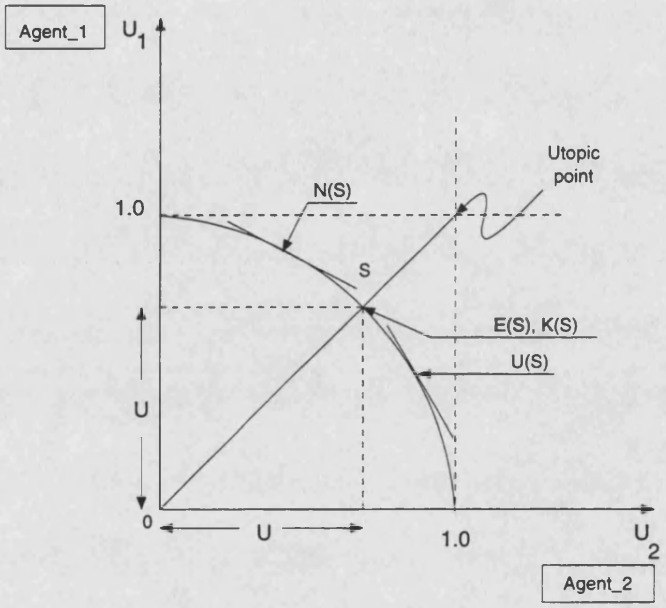


Figure 4.2: Joint course of action in game theory solutions

axes as asymptotes. The Nash solution in general evaluates an outcome which favours a particular agent with higher weight. However, there could exist special cases as shown by Kanappan and Marshek (1993) where each agent obtain the same individual pay-off.

- The Kalai-Somordinsky solution $K(S)$ [Kalai and Somordinsky (1975)] ensures equal pay-off to conflicting agents under normalised coordinates. For instance, as shown in Figure 4.2, for each utility level attained by Agent_1, the corresponding utility achieved by Agent_2 also increases. Therefore, an agent is not affected negatively by expanding the utility space. It is

on this key characteristic of expansion that $K(S)$ mainly rests. The presence of this non-negative influence is the main factor that distinguishes Kalai-Somordinsky solution from Nash solution. $K(S)$, under normalised coordinates thus represents a point on S such that $U_1 = U_2 = U$, which is the point of intersection of S and a line joining the origin and the utopic points.

- The Egalitarian solution $E(S)$ represents a point on S which is the point of intersection of S and a 45^o line to the ordinate. E(S) represents a variant of the Kalai-Somordinsky solution. E(S) evaluates equal utility gain to both agents *whether the coordinates are normalised or not.*

- The Utilitarian solution $U(S)$ represents a point on S such that the sum $[\Sigma_{i=1}^{2} U_i]$ is maximum. Unlike the above three solutions, the Utilitarian solution presents a drawback of multiple solutions. For example, there could exist more than one point which corresponds to a maxima. Hence the applicability of Utilitarian solution is generally restricted to problems with convex Pareto-optimal boundaries.

Due to the characteristic of equal utilities, the Kalai-Somordinsky solution is generally preferred among other solutions such as Nash or Utilitarian. The Utilitarian solution, similar to the Nash approach, favours one agent with respect to its individual pay-offs. These solutions apart from satisfying Pareto optimality must also satisfy the condition of symmetry. The condition of symmetry suggests that the solutions $N(S), K(S), E(S)$ and $U(S)$ remain the same if the agents are interchanged in the geometrical description of the problem in Figure 4.2. These game theory solutions are illustrated through an example next.

4.2.2 An example to demonstrate game theory methods

A design example of a pressure vessel is considered here as a two agent game where agents have individual preferences on a specific design attribute[1]. These preferences influence the agents' choice on a common design variable which lead to a conflict between the agents. The resolution of the conflict is demonstrated via the four game theory solutions presented in Section 4.2.1.

(a) Specification of pressure vessel design problem

Figure 4.3 shows a symmetric pressure vessel with hemi-spherical ends which is to be designed by two agents (A_1 and A_2) to withstand a pressure of $P =$ 800KPa. The inner radius of the vessel (r) is given as 1.8m and length (l) = 10m. The constraints are: wall thickness (t) $\leq 0.2m$, outer radius (x) $\leq 2.0m$ and $\sigma_{hoop} \leq \sigma_{allow}$[2]. The value for wall thickness (t) is to determined which would ensure safe design. The pressure vessel design equations are given below [Gere and Timoshenko(1991)]:

$$W = \rho[\frac{4}{3}\pi(r+t)^3 + \pi(r+t)^2 - (\frac{4}{3}\pi r^3 + \pi r^2 l)] \quad (4.1)$$

$$\sigma_{hoop} = \frac{Pr}{t} \quad (4.2)$$

[1]This example is derived from a similar case considered in Gere and Timoshenko (1991) and Rao, *et al.*, (1997).

[2]Only the hoop stress σ_{hoop} is considered. Due to the symmetry of the vessel, the shear stress is ignored.

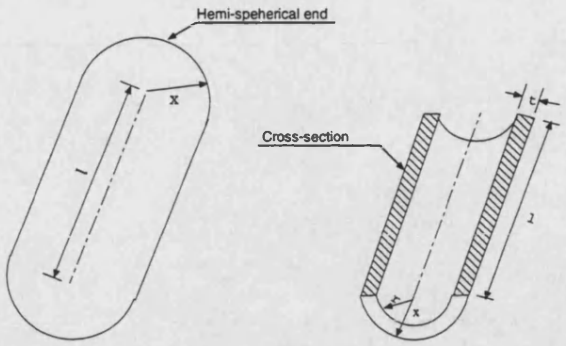


Figure 4.3: Schematic of a pressure vessel with hemi-spherical ends

(b) Formation of agent utility functions

As shown in Figure 4.3 the agents have different design preferences: A_1 wishes to minimise the weight of the vessel whereas A_2 wishes to minimise the hoop stress. Wall thickness (t) is a variable that is significant in the evaluation of weight (W) and hoop stress (σ_{hoop}) [Equations 4.1 and 4.2]. Therefore the agents control their individual preferences on W and σ_{hoop} by varying t. These preferences are shown in Figure 4.4 as utility functions with respect to t. Figure 4.4(a) shows the utility function for A_1 which indicates a maximum utility of 1.0 for a wall thickness of 0.05m. The utility curve shows a linear decrease for any increase in wall thickness (t) since the weight W of the vessel is directly proportional to t. Similarly variation pattern for A_2 shown in Figure 4.4(b) a maximum when wall thickness is 0.2m since wall thickness (t) is inversely proportional to the hoop stress (σ_{hoop}). The utilities shown in Figures 4.4(a) and (b) are thus consistent with the preferences of the agents. Based on the individual utility functions, a feasible space is obtained as shown in Figures 4.4(c) to check the existence of negotiated solutions, and the range of negotiation.

(c) Evaluation of game theory solutions

For the evaluation of game theory solutions to the conflict, a joint utility curve is plotted under normalised coordinates as shown in Figure 4.5 to obtain the game

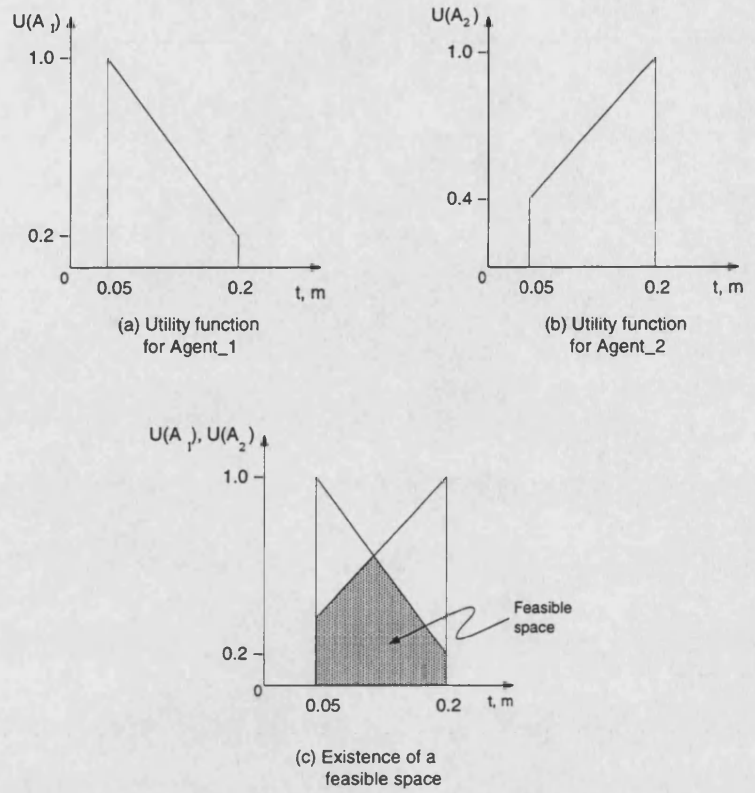


Figure 4.4: Utility functions for wall thickness

theoretic outcomes. These outcomes are also indicated in the figure. The Nash and Utilitarian solutions N(S) and U(S) yielded maximum benefit for $A_1(= 1.0)$ which corresponded to a value of 0.05m for wall thickness. These solutions yielded the minimum possible utility for agent $A_2(= 0.4)$. The Kalai-Somordinsky and Egalitarian solutions K(S) and E(S), however, benefited both agents equally [utility values = 0.7] and resulted in a value of 0.125m for the wall thickness.

The game theory solutions obtained in the context of the pressure vessel problem indicated how trade-offs are obtained in parametric design. The isolation of single negotiated solution from a set of four game theoretic solutions is discussed later in Section 4.3.4. There exist inter-dependencies between variables in design problems such as influence of wall thickness on hoop stress. The evaluation of dependencies are useful for rationalising the utility functions and is discussed next.

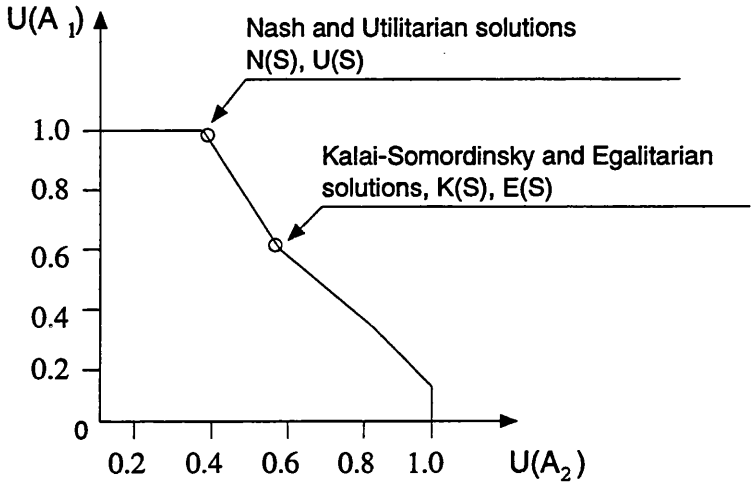


Figure 4.5: Game theory solutions for the pressure vessel problem

4.2.3 Dependency based reasoning

In dependency based reasoning, the domain variables are usually classified into several types according to their role in design problems. For example, some variables are critical to the design whereas some others may have marginal influence on the overall problem. Kusiak and Wang (1995) classified a group of design variables into five categories (decision, performance, intermediate, constraint and specification), of which the key types are *decision* and *performance* variables. The decision variables are the ones the agents control in order to reach or evaluate a performance variable. Thus in dependency-based negotiation, the problem is to determine the effect of certain decision variables on performance variables. Such an analysis involves two distinct aspects: qualitative and quantitative reasoning. The confluence based qualitative reasoning provides a means to identify the possible direction of change between decision and performance variables [de Kleer and Brown (1984), Kuipers (1986)]. The quantitative reasoning evaluates the actual change in the numerical value of a performance variable based on perturbation analysis on decision variables [Soo and Wang (1992), Kusiak and Wang (1995)].

A typical dependency network graph is shown in Figure 4.6 and can be expressed

as $G = \{V, E, \Omega, \Psi\}$ where V is the set of vertices, each representing a design variable and E is a set of directed edges between these vertices. Ω and Ψ are respectively the sets of qualitative and quantitative dependencies between the design variables [Wellman (1991)].

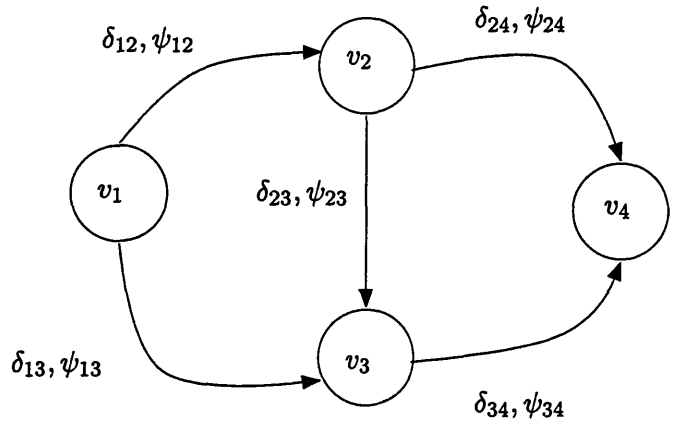


Figure 4.6: A sample network graph (G)

(a) Qualitative dependency

The qualitative dependency between two design variables v_1 and v_2 is symbolically represented as $\delta_{v_1,v_2} \in \Omega$, and is defined as follows:

$$\delta_{v_1,v_2} = \begin{cases} + & \text{if } \delta_{v_1} = +, \Rightarrow \delta_{v_2} = +, \textit{and} \\ & \text{if } \delta_{v_1} = -, \Rightarrow \delta_{v_2} = - \\ - & \text{if } \delta_{v_1} = +, \Rightarrow \delta_{v_2} = -, \textit{and} \\ & \text{if } \delta_{v_1} = -, \Rightarrow \delta_{v_2} = + \\ 0 & \text{if } \delta_{v_1} = +, \Rightarrow \delta_{v_2} = 0, \textit{and} \\ & \text{if } \delta_{v_1} = -, \Rightarrow \delta_{v_2} = 0 \\ ? & \text{Otherwise - case of ambiguity} \end{cases} \tag{4.3}$$

The qualitative reasoning in a dependency network is achieved by using parallel or serial dependencies [de Kleer and Brown (1984), Kuipers (1986), Kusiak and

Wang (1995)]. Figure 4.7(a) represents a parallel dependency which is expressed as $[\delta_{v_1,v_3} \oplus \delta_{v_2,v_3}]$ between the three variables v_1, v_2 and v_3. Figure 4.7(b) represents the serial dependency expressed as $[\delta_{v_1,v_2} \otimes \delta_{v_2,v_3}]$. The $\oplus$ and $\otimes$ are parallel and serial operators. This basic relationship expressing parallel or serial dependencies can be generalised to any network graph.

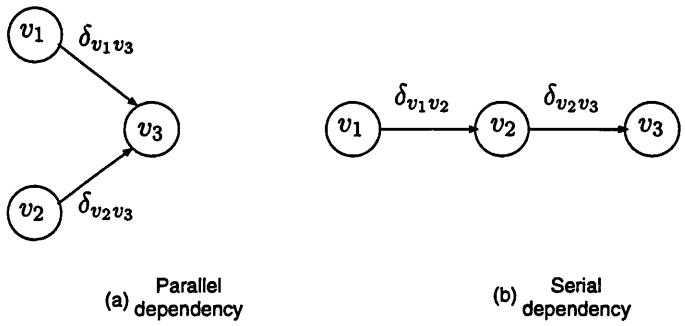


Figure 4.7: Parallel and serial dependency networks

(b) Quantitative dependency

In evaluating quantitative dependency between two design variables v_1 and v_2, it is usually assumed that an analytical relationship exists between the variables as given by,

$$v_2 = f(v_1) \tag{4.4}$$

Therefore the quantitative dependency between the variables v_1 and v_2 can be given as,

$$\psi_{v_1v_2} = \frac{\Delta v_2}{v_2^0} \frac{v_1^0}{\Delta v_1} \tag{4.5}$$

The relationship for $\psi_{v_1v_2}$ represents the quantitative change in v_2 by perturbing v_1 by r percent; v_1^0 and v_2^0 represent the initial or current values of v_1 and v_2. The principle of parallel and serial dependencies can be applied to the case shown in Figure 4.7(a) and (b) which would become $[\psi_{v_1,v_3} + \psi_{v_2,v_3}]$ and $[\psi_{v_1,v_2} \times \psi_{v_2,v_3}]$.

(c) Reduction of design networks

In a dependency network it is often useful to know the *reduced* or *net* dependency between any two variables. The reduction involves the removal of intermediate variables from the design dependency network retaining only the decision and performance variables. This is done to ease the evaluation of proportional change in one variable with respect to a change in the other. For example, for the network graph shown in Figure 4.8, the qualitative and quantitative dependencies $[\delta_{v_1,v_4}$, $\psi_{v_1,v_4}]$ are expressed as,

$$\delta_{v_1,v_4} = [\delta_{v_1} \otimes (\delta_{v_2} \oplus (\delta_{v_5} \otimes \delta_{v_4}))] \oplus (\delta_{v_3} \otimes \delta_{v_4}) \tag{4.6}$$

$$\psi_{v_1,v_4} = [\psi_{v_1} \times (\psi_{v_2} + (\psi_{v_5} \times \psi_{v_4}))] + (\delta_{v_3} \times \psi_{v_4}) \tag{4.7}$$

A general reduction procedure is presented in Section 4.3.2.

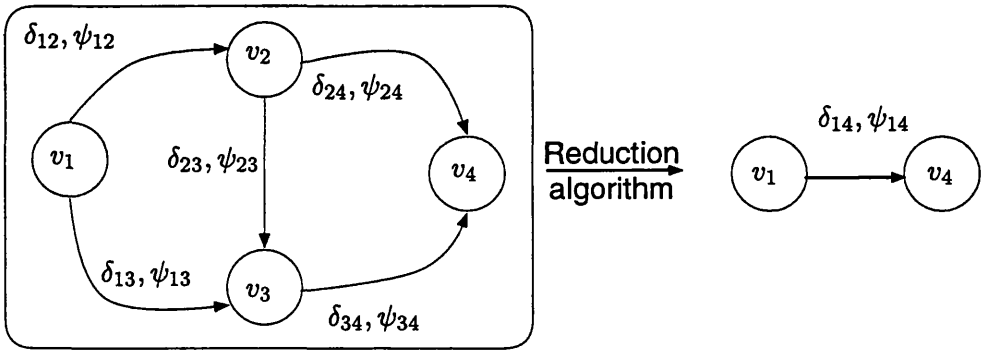


Figure 4.8: Network reduction process

4.2.4 An example to illustrate dependency based reasoning

In order to explain the qualitative and quantitative reasoning in a design network, a simple example of a block with a bore is considered[3]. The variables describing the geometry of the block are shown in Figure 4.9.

[3]For the sake of simplicity, constraints on volume of bore and block are not introduced except for the geometric constraint: bore diameter(d) < breadth of block (b) [Figure 4.9].

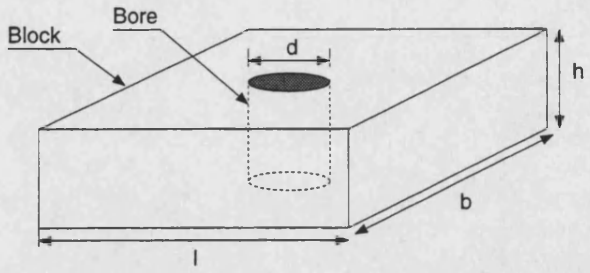


Figure 4.9: Schematic of a block with a central bore

(a) Construction of network graph

A network graph (G) is constructed for the block based on a given set of domain equations. In this case these equations are:

$$V_H = \frac{\pi d^2 h}{4} \tag{4.8}$$

$$V_B = lbh - V_H \tag{4.9}$$

$$V_B = lbh - \frac{\pi d^2 h}{4} \tag{4.10}$$

where h = height of the block, V_H = bore volume, l = length and V_B = volume of the block.

Each variable in the analytical expressions Equations 4.8 and 4.9 represent the vertices (or nodes) of a directional graph (G). These vertices are shown in Figure 4.10(a). Based on the analytical relations, edges are drawn between the vertices as shown in Figure 4.10(b). For example, variable d appears in the expression for V_H. Therefore an edge is drawn from vertex d to vertex V_H. The same principle is applied to obtain the complete network in Figure 4.10(b). This presents the construction of the dependency graph. Each edge (E) connecting two vertices in the graph has its associated qualitative and quantitative dependencies. Their evaluation is presented next.

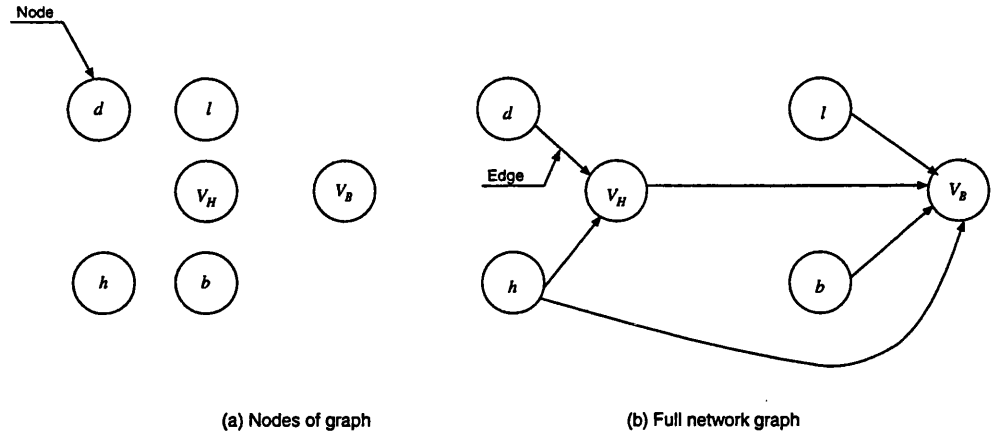


Figure 4.10: Network for the block example

(b) Qualitative dependencies

The volume of bore (V_H) and volume of block (V_B) are considered to examine the qualitative influences. Figure 4.11 shows the influences on V_H and V_B separately. As shown in Figure 4.11(a), the influence of bore diameter d on V_H is '+' since in Equation 4.8, increasing d will show a corresponding increase in V_H assuming there are no constraints on h. Therefore a '+' sign is assigned to the qualitative influence δ_{d,V_H}. Similarly, from Equation 4.10, the qualitative influence δ_{d,V_B} of d on V_B is '−' since increasing d will decrease V_B. The influence of 'l' and 'b' on V_H are '0' since l and b do not appear in Equation 4.8 for V_H. Thus $\delta_{l,V_H} = \delta_{b,V_H} = 0$.

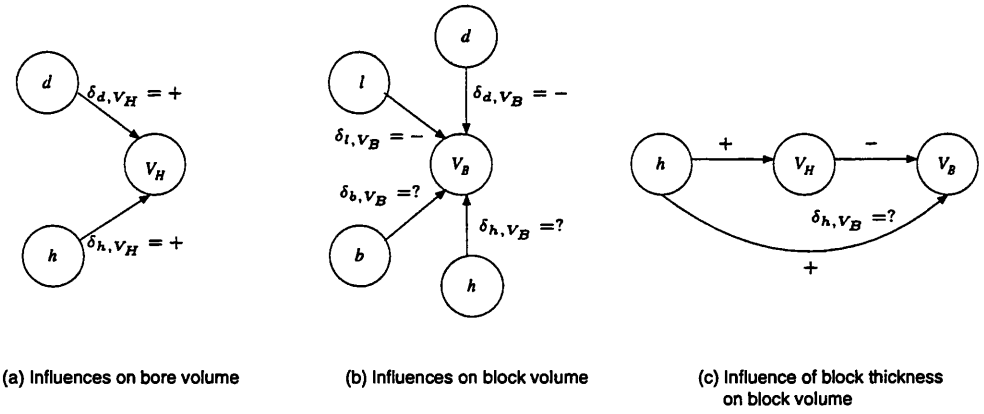


Figure 4.11: Sub-network for the block example

To demonstrate the case of ambiguity represented as '?' in Equation 4.11, the influence of h to V_B is considered as shown in Figure 4.11(c). The direct and

indirect influences of h on V_B need to be combined to obtain the effective qualitative influence δ'_{h,V_B}. A symbolic form of representation of Figure 4.11(c) is:

$$\begin{aligned} \delta'_{h,V_B} &= (\delta_{h,V_H} \otimes \delta_{V_H,V_B}) \oplus \delta_{h,V_B} \\ &= (+ \otimes -) \oplus + \\ &= (- \oplus +) \\ &= ? \end{aligned} \tag{4.11}$$

Equation 4.11 suggests that there are both '+' and '−' parallel influences of h on V_B. Therefore, δ_{h,V_B} represents a case of ambiguity and is assigned '?'. In engineering problems, the qualitative dependencies alone are not sufficient for reasoning and therefore the quantitative dependencies between design variables also need to be determined to resolve qualitative ambiguities.

(c) Quantitative dependency

To illustrate the effect of quantitative dependencies, the block in Figure 4.12 is again considered. To enable the demonstration of the concept, some values are assigned to the geometrical variables as indicated in Figure 4.12. The various quantitative dependencies are:

$$\begin{aligned} \psi_{d,V_H} &= \frac{\Delta_{V_H}}{V_{H^0}} \frac{d^0}{\Delta_d} = 1.0 \\ \psi_{h,V_H} &= \frac{\Delta_{V_H}}{V_{H^0}} \frac{h^0}{\Delta_h} = 1.0 \\ \psi_{V_H,V_B} &= \frac{\Delta_{V_B}}{V_{B^0}} \frac{V_{H^0}}{\Delta_{V_H}} = -0.067 \\ \psi_{h,V_B} &= (\psi_{h,V_H} \times \psi_{V_H,V_B}) + \psi_{h,V_B} = +0.33 \end{aligned} \tag{4.12}$$

Thus the ambiguity '?' in the qualitative reasoning for δ'_{h,V_B} in Equation 4.11 can

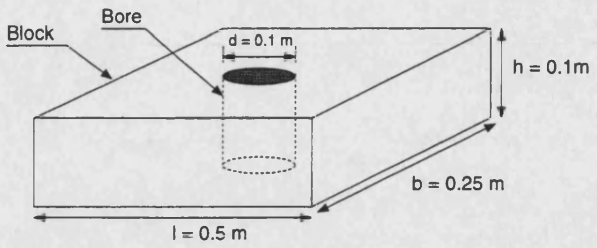


Figure 4.12: Quantitative values assigned to geometric variables of the block

be eliminated by the corresponding quantitative dependency ψ_{h,V_B} from Equation 4.12. In this case the influence of h on V_B is resolved to '+'.

(d) Network reduction

It is always convenient to express dependencies in the reduced form. For Figure 4.10, and the nominal dimensions in Figure 4.12, the effective dependencies between the variables h and V_B are therefore obtained as follows:

$$\psi_{h,V_B} = (\psi_{h,V_H} \times \psi_{V_H,V_B}) + \psi_{h,V_B} = +0.33 \quad (4.13)$$

$$\delta_{h,V_B} = (\delta_{h,V_H} \otimes \delta_{V_H,V_B}) \oplus \delta_{h,V_B} = \text{'+'} \quad (4.14)$$

The above equations represent the effective dependencies for a simple network graph in Figure 4.10. A general network reduction procedure developed for this study is presented in section. Having presented the game theory and dependency based reasoning schemes, their comparison is presented in the next section.

4.2.5 A comparison of game theory and dependency based reasoning techniques

This section presents a comparison of the two approaches to conflict negotiation based on certain factors[4] that influence a conflict situation. Figure 4.13 shows these factors and analyses their role in game theory and constraint relaxation techniques.

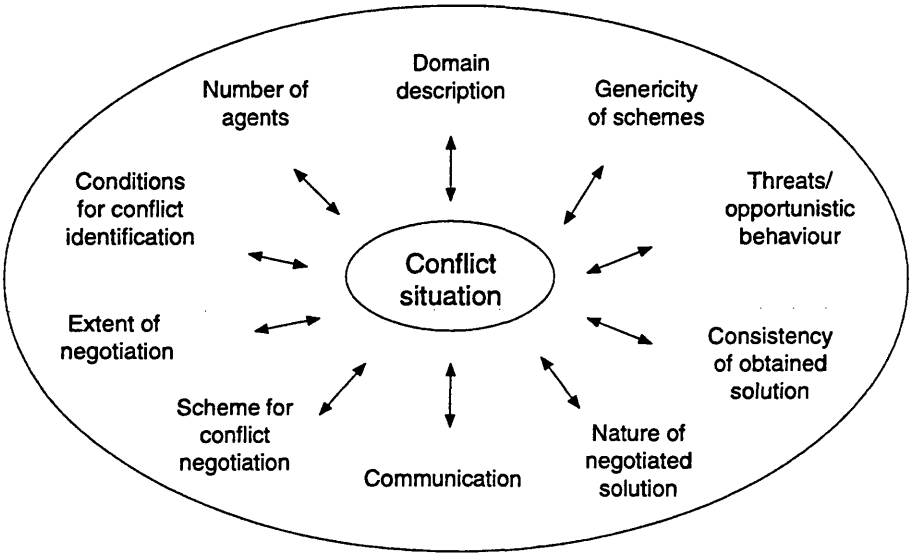


Figure 4.13: Factors influencing a conflict situation

Domain description

In engineering, the analytical formulae are often used to characterise the relationship between design variables in a domain. An approximate relation between design variables (e.g., using response surface methodology) often leads to conflicts with respect to its idealised values. In game theory based techniques the domain is described using well-defined analytical formulae, whereas in dependency based constraint relaxation, a network graph is used to represent the relationship between design variables. Parametric classification is also used in conjunction with these approaches as a means to separate the critical variables from the non-critical ones.

[4]These factors have been identified from various isolated sources including that of Klien (1989), Badrinath and Rao (1996), Nash (1950) and Rao (1999).

Number of agents

Although neither the game theory nor the constraint relaxation approach poses any restriction on the number of agents, the former has mainly been studied in the context of two-player games. In some isolated cases game theory based methods have been extended to address n-agent bargaining by means of coalition building techniques [Rao (1999)]. Building such coalition and a unified mathematical function representing a *common coalition objective* is often difficult due to the restrictions imposed by the Arrow's impossibility theorem [5]. However, if such common objectives are collectively stable, then game theory would yield negotiated solutions (e.g., as in wage bargaining). But due to the inability in guaranteeing unconditional stability, such coalition formations are not generic. In the case of fully cooperative conflicts, the dependency based constraint relaxation schemes have been applied to cases where the number of conflicting agents is more than two [6] [Kusiak and Wang (1996)].

Condition for conflict identification

Appropriate well-defined conditions for conflict detection are necessary to develop efficient negotiation schemes. These conditions include design constraint violation and variation in preferences between agents. Constraint relaxation approaches check for the violation of any design constraints. In game theory based methods, conflicts are identified when there is a difference between the individual preferences of agents on some shared design attribute.

---

[5] Arrow's impossibility theorem [Arrow (1964)] states that it is always not possible to construct group utility functions representing preferences of a group. Suppose there are three alternatives A_1, A_2 and A_3 and their preferences are stated as $A_1 \succ A_2$, $A_2 \succ A_3$, $A_3 \succ A_1$ ($\succ$ indicates *is preferred to*). As can readily be seen, in a typical bargaining situation such group preferences could stay *ad infinitum* in a loop. Arrow argued that any coalition that expresses such a preference structure is irrational and intransitive; irrational due to the fact that the group may choose to keep the cyclic preferences just to stay in the loop. As a consequence, any negotiation process that portrays intransitive preferences is doomed to failure.

[6] However, it should be emphasised that the way Kusiak and Wang (1996) treated conflict situation differs in important ways from that considered by the author; the key difference being the fact that in author's research conflicts occur due to difference in preferences whereas Kusiak and Wang presented conflicts as violation of design constraints. In this sense, Kusiak and Wang eliminated the restrictions on preference-formation imposed by Arrow (1964).

Extent of cooperation

Full cooperation in negotiation means that any point within the feasible space represents a solution. However, usually agents do have preferences and total cooperation seldom exists in a decision process. Dependency based constraint relaxation does not provide any direct means for representing agent preferences. The game theory based Nash or Kalai-Somordinsky techniques offer means for obtaining a solution which is usually not the ideal one (shown in Figure 4.2 of Section 4.2.1 as *utopia or utopic point*). Therefore game theory can be applied to situations where total cooperation between agents cannot exist due to their set preferences.

Scheme for conflict negotiation

The main approaches are based on game theory and dependency based reasoning. There are, however, variations from these such as informal constraint relaxation approaches, for example, Berker and Brown (1996), or the use of *if-then* rules. There is no definite basis to favour a particular approach: choice of the approach is dependent on the conflict situation (e.g., conflicts in parametric design).

Threats and opportunistic behaviour

The presence of threats and opportunistic behaviour cannot be ruled out in a conflict situation, particularly in non-cooperative conflicts. While the constraint relaxation based approach does not take into account such behaviour, some game theoretic techniques can include weighting techniques to model threats in bargaining. In game theoretic methods these are often identified as the threat value of an agent which can be used to influence other agents to reach an agreement. However, rational quantification of threats is required to include their consideration within a formal method.

Genericity of schemes

Game-theoretic approach can be applied to problems with well defined analytical formulae or some mathematical representation such as the use of utility functions.

The genericity of a game-theoretic approach has been tested outside the engineering domain [e.g., wage-bargaining, international business negotiation, etc.]. The constraint relaxation based approach is generic enough to be extended to any domain of conflict. However, this is subject to the existence of exact or approximate relationships between design variables.

Communication

Communication is central in any coordinated group activity. In both the approaches communication could take place in an informal manner or via some formal language such as KQML requiring formal semantics [Finin, *et al.*, (1995)]. Formal mechanisms for communication are preferred to avoid inaccuracies in information exchange, for better understanding between the conflicting agents and possibilities of secondary conflicts.

Nature and consistency of negotiated solutions

This is one of the key factors that influence a conflict. In the constraint based approach, the feasibility space bounded by rationality[7] leads to a set of solutions acceptable to all conflicting agents. Hence in a constraint relaxation approach there could be a typically large feasible set for a given conflict problem. On the other hand, the game theoretic solutions lead to single-point solutions within the feasible space bounded by each agent's preferences. Though game solutions such as Nash and Kalai-Somordinsky solutions could lead to several single point solutions, the number of solutions is finite as compared to an infinite set generated by constraint-based approaches.

In the constraint relaxation approach [Figure 4.14], assuming that all perspectives fully co-operate, each perspective (P_i) proposes a solution (S_{ij}). A negotiated solution exists only if $S_1 \cap S_2 \cap ... \cap S_n \neq \{\phi\}$. When such a non-empty set is formed, it consists of all feasible alternatives for the design [feasible space in

[7]Rationality in this context refers to an agent's ability to coordinate its actions to its goals in a problem bounded by design constraints. Rationality thus characterises the choice of an agent's action based on a given set of constraints.

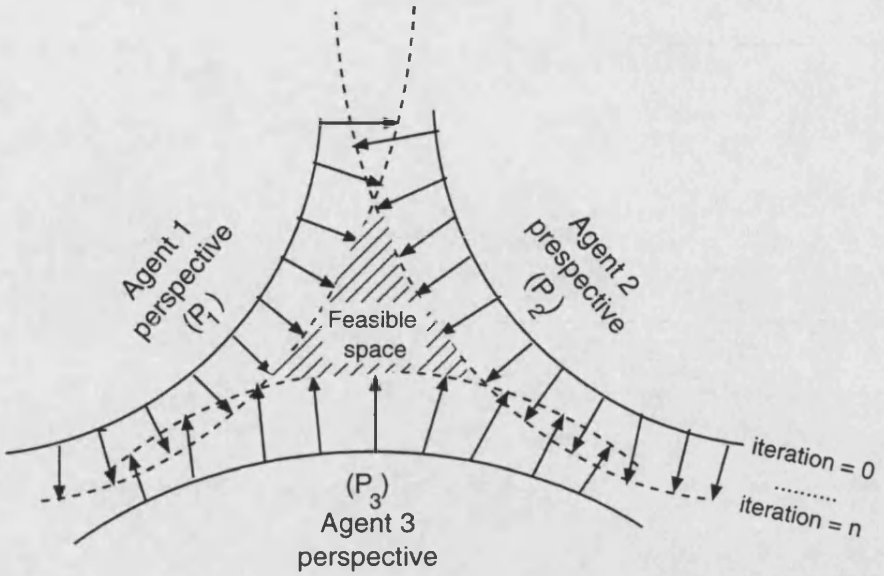


Figure 4.14: Constraint relaxation process

Figure 4.14]. To form a feasible space one has to go through several iterations and the convergence of the solution cannot be guaranteed. But assuming that such a set is evaluated, no efficient methods are present for isolating a single solution as compared to single-shot game-theoretic methods.

The constraint relaxation approach as presented by Kusiak, *et al.*, (1996) and Kersten (1988), however, offers a weak method of contraction to isolate a single solution as shown in Figure. The process of contraction begins by gradually relaxing the aspiration levels of each perspective by some small increments (ϵ). This process is continued until a single point solution is reached [Figure 4.15]. The extent of relaxation from each perspective is entirely governed by the aspiration level[8] of the perspective and the degree of cooperation between perspectives. For example, in a fully cooperative process, P_1 may choose to relax its aspiration levels whereas P_2 and P_3 may not choose to relax their aspirations at all. Thus

[8] Aspiration level indicates a range within which a constraint may be relaxed

the process of contraction of feasible space immediately presents two drawbacks: (a) this is an additional iterative process requiring to check for consistency of feasible space for every iteration and (b) convergence to a single solution cannot be guaranteed at all times.

Generally in negotiation, full co-operation between perspectives should not be assumed, as pointed out by Davey and Olson (1998). This is due to the fact that negotiating trade-offs is critical in processes involving multiple agents as conflicts due to individual preferences are inevitable. The contraction technique (shown in Figure 4.15) is further weakened when agents belonging to each perspective have individual preferences.

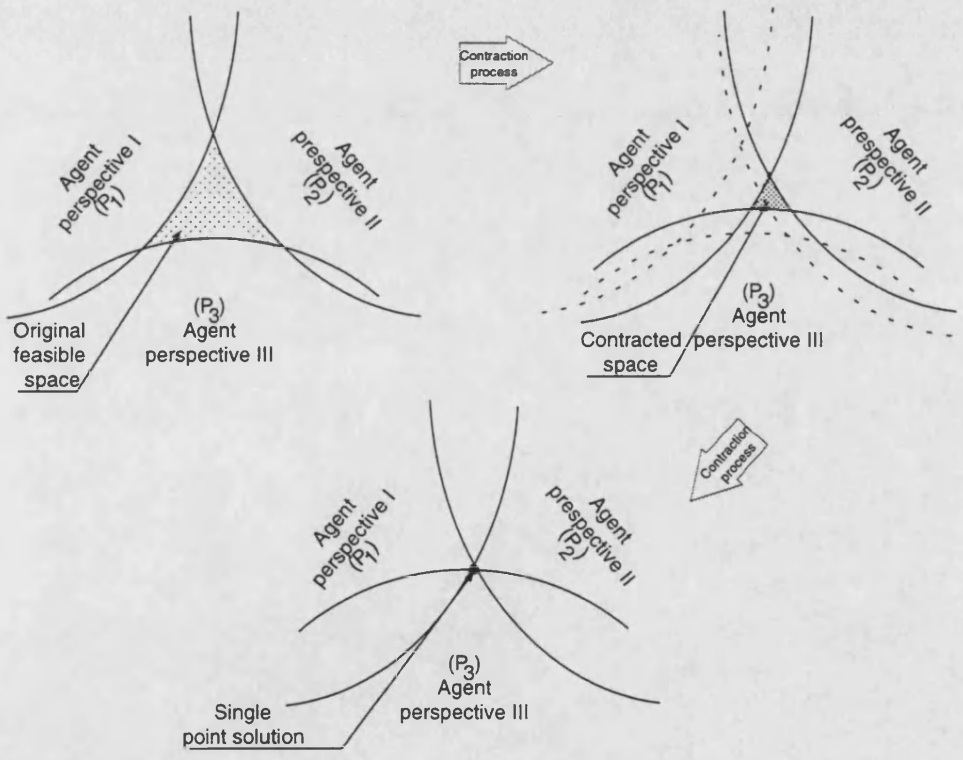


Figure 4.15: Contraction process

The constraint relaxation technique discussed by Kusiak, *et al.*, (1996) does not present a scenario where co-operation between perspectives could only be partial. Another aspect where the two approaches (ie., game theory and constraint relaxation) seem to differ is the condition assumed for identification of conflicts.

In the constraint relaxation approach, constraint violation is often imposed as the main condition for conflict detection [for example, see Kusiak, *et al.*, Berker and Brown (1996)]. From a negotiation perspective this is a weak condition especially where preferences are allowed to exist within the Pareto space. In other words, constraint violation could act as an additional condition for reaching an agreement between conflicting entities. In so doing, the central role played by the agents in negotiation will be consistent with the real world situations.

The game theoretic approaches evaluate a single point negotiated solution but the effect of dependencies between design variables is generally ignored. This could lead to localised solutions as stated by Kusiak *et al.*, (1996). In order to maintain a global consistency in the negotiated solutions the use of qualitative and quantitative reasoning in directed network graphs assist in propagating the effect of parametric deformation uniformly throughout the domain. In addition, the use of aspiration levels relates well to the real world scenario where decision makers change their preferences in the midst of a negotiation process[9]. In most of the game-theory based approaches the utility variation for each agent involved in the conflict is held constant. This decreases the flexibility of the negotiation process and could also affect the formation of a combined utility space between the conflicting agents.

For effective conflict negotiation in product development, there is a need to combine dependency-based constraint relaxation with game-theoretic approaches. Such a combination would be necessary to develop efficient conflict resolution schemes with consistent solutions in an agent-supported collaborative process. This is the object of the next section.

[9]This view is also well supported in the literature by several researchers [Davey and Olson (1998), Sycara and Lewis (1991), Kersten (1988)].

4.3 A scheme for agent-supported conflict resolution

A comparison of game theory and constraint relaxation approaches emphasised the need for consistent design solutions. Therefore a conflict resolution scheme with the advantages of the two approaches could be considered beneficial. Such a scheme would capture the following key attributes:

- evaluation of dependencies
- agent's preferences
- single point solution

Each agent's preferences need to be explicitly stated for negotiation to be realistic. This is carried out by specifying certain utility functions for each agent involved in a conflict. A single-shot game-theoretic technique would yield a negotiated settlement. Based on the global constraint network the effect of negotiated solution is propagated throughout the domain to ensure the obtained solutions are consistent.

4.3.1 Identification of conflicts

A collaborative process (P_c) consists of expert agents (A_i, $i = 1....n, n \in \Re$). Each agent A_i corresponds to a perspective (P_j, $j = 1....m, m \in \Re$) e.g., design, manufacturing and marketing stages in product life-cycle design. Expert agents associated with each perspective ($A_i \equiv P_j, m = n$) are responsible for task processing. Each perspective consists of several design variables V such that $V \subset (A_i \equiv P_j)$. An abstract view of the negotiation process is shown in Figure 4.16.

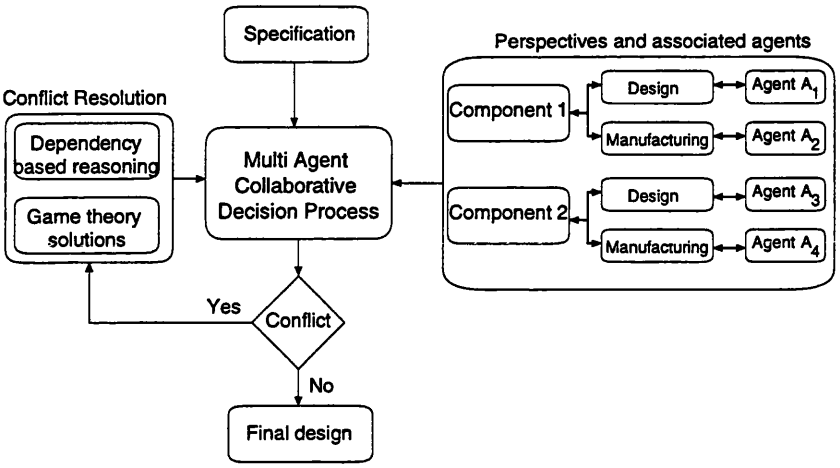


Figure 4.16: An abstract view of a negotiation process

- The problem specification is usually captured as a set of design variables $V_S \in V$ which are assigned values at the start of the design process. This set of variables contains certain common requirements which are common to all agents.

- Each agent (A_i) in the decision process is assumed to have a set of preferences U^{A_i} over a set of shared variables $V_{SH} \in V$ such that $U^{A_i} \neq \{\phi\}$.

- The evaluation of unknown variables including the decision $[V_D \in V]$ and performance variables $[V_P \in V]$ are carried out and a initial design is formed.

- Each agent (A_i) in a decision process controls certain key decision variables which are also directly related to the performance variables.

- As the agents interact to examine the acceptance of the initial design, a conflict could be detected due to differences in preferences over shared variables, [i.e., $A_i(V_{SH}) \neq A_j(V_{SH}), i \neq j$].

A new scheme has been developed which combines game theory and dependency based reasoning in order to resolve such conflicts [conflict type identified in Section 2.5.2, Figure 2.4]. A detailed reasoning procedure is presented next which is used to evaluate the effective qualitative and quantitative dependencies.

4.3.2 Reasoning on a design network

A design network graph ($\mathbf{G}$) is formed based on the analytical formulae [of the form $f(V)$] which describe the domain. Such a network is useful in propagating any changes throughout the domain. This network provides quantitative (ψ) as well as qualitative (δ) dependency information between any two design variables. The Figure 4.17 shows a directed graph $\mathbf{G}$ represented as a four tuple $\mathbf{G} = (V^e, E, \Omega, \Psi)$; $\delta \in \Omega$ and $\psi \in \Psi$, where:

E is the set of directed edges.

δ is the qualitative dependency of each edge (E).

ψ is the quantitative dependency of each edge (E).

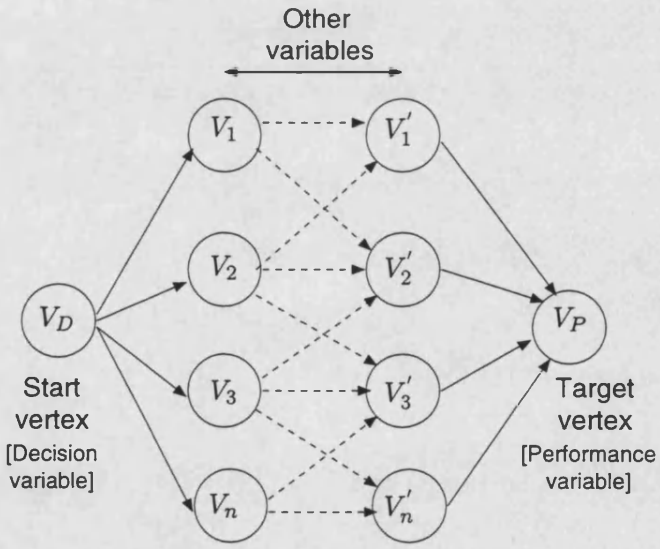


Figure 4.17: Graph (G) showing the start (V_D) and the target (V_P) vertices

When a full network is specified it has a non-empty set of edges, vertices and dependencies. Of immediate interest are the decision and performance variables and the link between them. Thus the aim is to establish effective dependencies between the decision and performance variables. A reasoning procedure `reduce_network` is outlined as follows:

Begin

Procedure reduce_network

$\{\Omega\} = \{\phi\}$; initialise all variables

$\{\Psi\} = \{\phi\}$;

$\forall V_P \in V$ and $V_D \in V$, specify a pair: (V_P, V_D)

$V_D = N_{start}$

$V_P = N_{end}$

for each V_D, create P_{INFO}

$P_{INFO} = \{C_N, C_P, C_{PL}, \psi\}$

push $P_{INFO} \rightarrow I_{NS}$ stack

while $I_{NS} \neq \{\phi\}$, do

$C_I \leftarrow$ pop (I_{NS})

retrieve from C_I; C_N, C_P, C_{PL}, ψ

for each Adj(C_N), do

$A_N \leftarrow$ Adj(C_N)

$A_{NPL} \leftarrow$ Adj$(C_{PL} + 1)$

$C_P(A_N) \leftarrow C_P +$ Adj(C_N)

if $A_N = V_P$, then

$\delta_i = \delta_{V_D,V_1} \otimes \delta_{V_D,V_2} \ldots\ldots\ldots \otimes \delta_{V_n,V_P}$

$\psi_i = \psi_{V_D,V_1} \times \psi_{V_D,V_2} \ldots\ldots\ldots \times \psi_{V_n,V_P}$

store $\{C_P, C_{PL}, \delta_i, \psi_i\}$ $(i = 1 \ldots n, n \in \Re)$

else

new $P_{INFO} = \{$Adj$(C_N), AN_{PL},$ Adj$(C_N)_{path}, \psi\}$

push $P_{INFO} \rightarrow I_{NS}$ stack

end if

end do

end for

end for

end for

∀ stored paths, evaluate effective dependencies,

$\delta_{eff} = \delta_1 \oplus \delta_2 \oplus ..... \oplus \delta_k$

$\psi_{eff} = \psi_1 + \psi_2 + ... + \psi_k$

`end for`

End

Given a source vertex (which is usually a decision variable, V_D) and a target vertex (which is a performance variable, V_P) the first step towards reduction is to identify all paths from V_D to V_P. Path identification is carried out by creating a path information object P_{INFO} which consists of the current node (C_N), the current path (C_P), the current path length (C_{PL}) and the corresponding quantitative dependencies (ψ). For example, if $C_N = V_D$, then the values for C_P, C_{PL}, and ϕ are $(V_D, 0, 0)$ respectively. The P_{INFO} object contains information including the path traversed at any instant from the start node and the corresponding path length by keeping track of the number of paths traversed. Each time a P_{INFO} is formed, it is pushed into a stack waiting to be retrieved. Thus each stack item is unique in terms of its contents.

Whenever a P_{INFO} object is popped out of the stack, its contents are extracted and for each $C_N \in P_{INFO}$ the adjacencies `Adj`(C_N) are determined. Since the procedure `reduce_network` is a modification of the depth-first search technique, the effective dependency of each path is obtained as a product of dependencies. Once all possible paths between the source and the target are determined, the final effective dependency is obtained as the sum of individual path dependencies.

4.3.3 Example to demonstrate the reasoning procedure

A sub-set of the design example of block with a helical spring [adapted from Juvinall and Marshek (1991)] is considered here to demonstrate the construction

of network from domain equations and reasoning procedure. A helical spring with squared and ground ends is required to exert a certain force F with constraints on length. The designed spring must fit inside a bore of a rectangular block with constraints on bore diameter as shown in Figure 4.18.

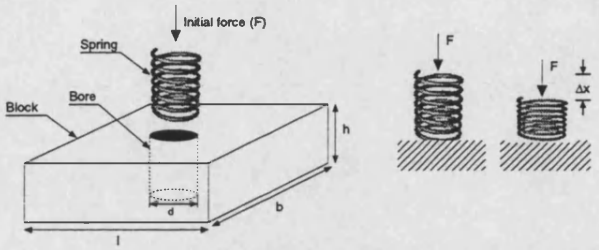


Figure 4.18: Rectangular block with a spring

The steps involved in reasoning procedure (reduce_network) is systematically presented in the context of the design example. Based on the analytical design equations [given in Appendix A], a directed network graph as shown in Figure is generated. The focus is on the dependencies between the decision and performance variables. For instance, let initial force (F) be a decision variable and constraint (H_1) on free length be a performance variable. For the sake of clarity a sub-network of variables connecting F and H_1 is shown in Figure 4.19. The first stage is to create a P_{INFO} object (explained in section 4.3.2) consisting of the start node, the path traversed so far, the path length and the quantitative dependency. As indicated in the Figure 4.20, P_{INFO} is updated constantly and a path is retrieved as soon as the target vertex is reached. During the traversal process, adjacencies of each node are stored separately for further processing. For the case shown in Figure 4.19 the total number of paths was found to be four.

Once all paths between the start and the end vertex are known, the corresponding effective dependencies are evaluated as follows:

For PATH 1:

$$\delta = \delta_{F,H_1} \tag{4.15}$$

$$\psi = \psi_{F,H_1} \tag{4.16}$$

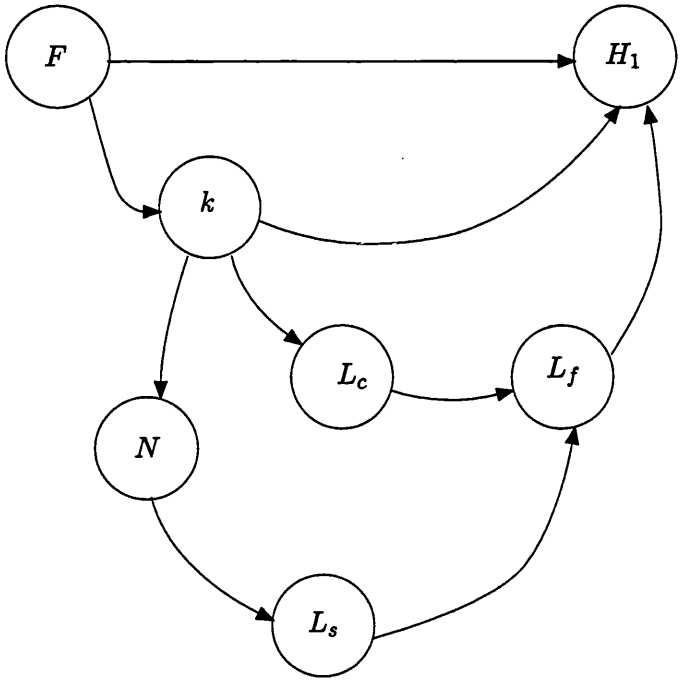


Figure 4.19: Network graph representing solid force and a constraint in the block-spring design problem

For PATH 2:

$$\delta = \delta_{F,k} \otimes \delta_{k,H_1} \tag{4.17}$$

$$\psi = \psi_{F,k} \times \psi_{k,H_1} \tag{4.18}$$

For PATH 3:

$$\delta = \delta_{F,k} \otimes \delta_{k,L_c} \otimes \delta_{L_c,L_f} \otimes \delta_{L_f,H_1} \tag{4.19}$$

$$\psi = \psi_{F,k} \times \psi_{k,L_c} \times \psi_{L_c,L_f} \times \psi_{L_f,H_1} \tag{4.20}$$

For PATH 4:

$$\delta = \delta_{F,k} \otimes \delta_{k,N} \otimes \delta_{N,L_s} \otimes \delta_{L_s,L_f} \otimes \delta_{L_f,H_1} \tag{4.21}$$

$$\psi = \psi_{F,k} \times \psi_{k,N} \times \psi_{N,L_s} \times \psi_{L_s,L_f} \times \psi_{L_f,H_1} \tag{4.22}$$

Effective dependency:

$$\begin{aligned}\delta_{eff} &= \delta_{F,H_1} \oplus (\delta_{F,k} \otimes \delta_{k,H_1}) \\ &\oplus(\delta_{F,k} \otimes \delta_{k,L_c} \otimes \delta_{L_c,L_f} \otimes \delta_{L_f,H_1}) \\ &\oplus(\delta_{F,k} \otimes \delta_{k,N} \otimes \delta_{N,L_s} \otimes \delta_{L_s,L_f} \otimes \delta_{L_f,H_1}) \end{aligned} \tag{4.23}$$

OR

$$\begin{aligned}\delta_{eff} &= \delta_{F,H_1} \oplus (\delta_{F,k} \otimes \delta_{k,H_1} \\ &\oplus((\delta_{k,L_c} \otimes \delta_{L_c,L_f}) \oplus (\delta_{k,N} \otimes \delta_{N,L_s} \otimes \delta_{L_s,L_f})))) \\ &\otimes\delta_{L_f,H_1}) \end{aligned} \tag{4.24}$$

$$\begin{aligned}\psi_{eff} &= \psi_{F,H_1} + (\psi_{F,k} \times \psi_{k,H_1}) + (\psi_{F,k} \times \psi_{k,L_c} \times \\ &\psi_{L_c,L_f} \times \psi_{L_f,H_1}) + (\psi_{F,k} \times \psi_{k,N} \times \\ &\psi_{N,L_s} \times \psi_{L_s,L_f} \times \psi_{L_f,H_1}) \end{aligned} \tag{4.25}$$

OR

$$\begin{aligned}\psi_{eff} &= \psi_{F,H_1} + (\psi_{F,k} \times \psi_{k,H_1} + ((\psi_{k,L_c} \times \psi_{L_c,L_f}) + \\ &(\psi_{k,N} \times \psi_{N,L_s} \times \psi_{L_s,L_f}))) \times \psi_{L_f,H_1}) \end{aligned} \tag{4.26}$$

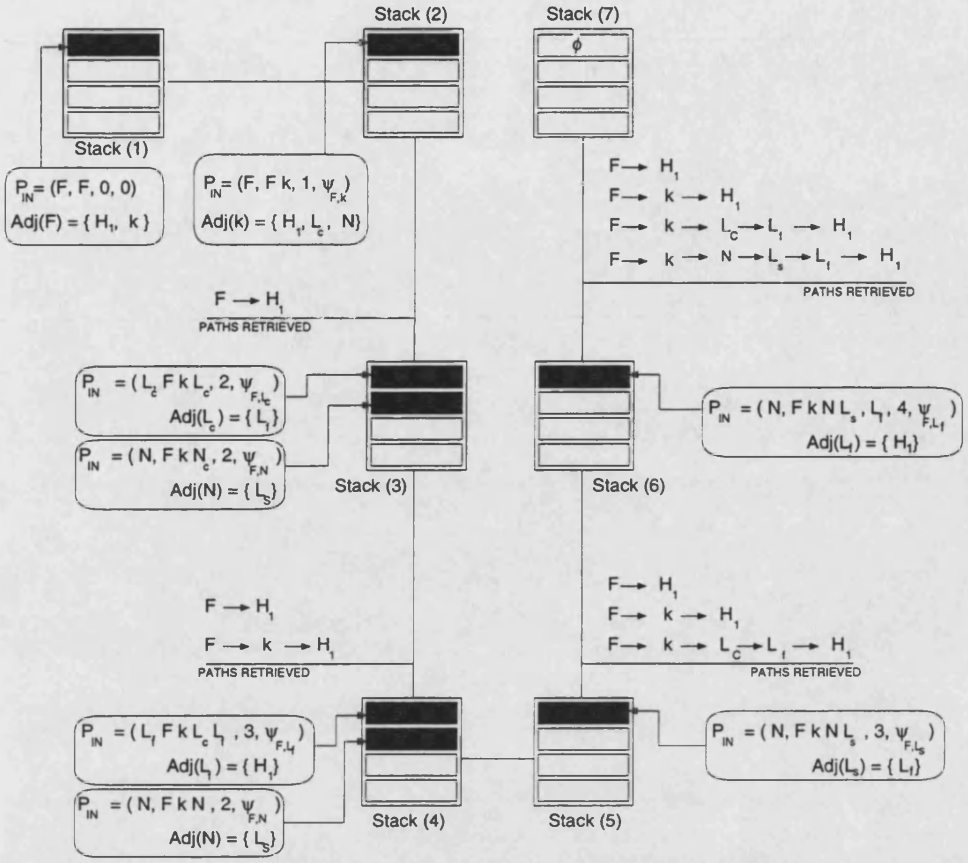


Figure 4.20: Path identification process

4.3.4 A trade-off mechanism to enforce decisions

The reasoning procedures evaluate the qualitative as well as quantitative influence of decision variables on performance variables in a design network. Such evaluation coupled with problem constraints assist the formation of utility functions for the agents. For example, in the pressure vessel design problem considered in section, the utility functions for the agents had to satisfy certain stress and geometric constraints of the problem[10].

[10]Based on literature in game theory related negotiation, there is no known standard way for constructing utility functions. For example, Davey and Olson (1998) criticised the view of utility based approaches and questioned the generality in assigning of weights in the generation utility functions. Several authors including Raiffa (1972) mentioned that the utility functions will need to pass through some test for rationality whereas other researchers as in Kanappan and Marshek (1993) have assumed some functions without any description on their validity. In the present research (as will be seen in Chapter 6) the design constraints coupled with reasoning procedures are used to guide the formation of utility functions.

Based on the prescribed utility functions for a given problem, using the Nash $N(S)$, Kalai-Somordinsky $K(S)$, Egalitarian $E(S)$ and Utilarian $U(S)$ methods, a one-shot compromise is reached resulting in four different game-theoretic solutions in general. However, only one solution out of the four can be chosen as a final negotiated settlement (N). The need for isolating a single solution is due to two reasons: (a) The pay-offs for each agent $(U_1^{A_1}, U_2^{A_2})$ obtained by the schemes are not always the same. Therefore, an agent may wish to choose a solution which yields highest possible utility to itself but not necessarily to other agents. (b) To prevent secondary conflicts which could delay the negotiation process. In the case of human conflicts, Nash (1950) suggested the use of a generic mechanism for breaking deadlocks in bargaining. Therefore, the following IF-THEN rules have been developed for enforcing decisions:

---

while S $\neq \{\phi\}$, do,
  if $U_1^{A_1} = U_2^{A_2}$ is true
    then final solution $N = \max(U_1^{A_1}, U_2^{A_2})$ in $\{N(S), K(S), E(S), U(S)\}$
  else if $U_1^{A_1} \neq U_2^{A_2}$
    then final solution $N = \min |U_1^{A_1} - U_2^{A_2}|$ in $\{N(S), K(S), E(S), U(S)\}$
  else N = Average $N(S), K(S), E(S), U(S)$ such that
    $N \subseteq$ S and $\{U_1^{A_1}, U_2^{A_2}\} \neq \{\phi\}$
  end if
end do

In the agent supported design of pressure vessel considered in Section 4.2.2, four game theory solutions (shown in Figure 4.5 were evaluated but acceptance of a single solution was not discussed. The IF-THEN rules presented above when applied to the pressure vessel design problem yields the Kalai-Somordinsky and Egalitarian as the final negotiated solution N (corresponds to equal utility values $U_1^{A_1}, U_2^{A_2} = 0.7$ for both agents).

The game theory methods, reasoning procedures and rules presented in sections are employed in developing reasoning and negotiation agents which is discussed next.

4.4 Software agents for reasoning and negotiation

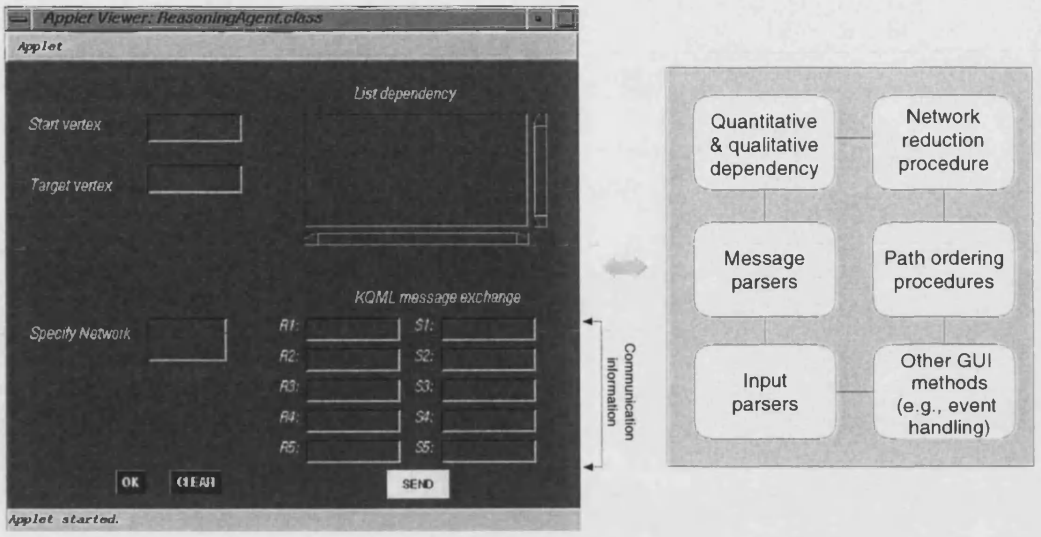
This section presents the software agents custom-developed for this study, viz., the reasoning and negotiation agents. To this end this section will cover the internal details of the agents such as their internal structure and the graphical user interface.

4.4.1 Reasoning agent

The reasoning agent [Figure 4.21] evaluates the dependencies between variables in a design network. First, the dependency network is input to the agent as sets of vertices, edges and quantitative dependencies in a directed graph as `<vertex>` `<edge>` `<quantitative_dependency>`. The reasoning agent forms the network graph and stores all the vertex and edge information.

The transition from SiFAs to the formation of the reasoning agent is shown in Figure 4.22. The key SiFAs and SimFAs in the reasoning agent are also shown in the figure. As indicated in Figure 4.22, the reasoning agent uses the network reduction procedures which was presented in Section 4.3.2 and evaluates the dependencies between all specified decision and performance variables.

The reasoning agent is generic and not dependent on a particular domain. The reasoning agent offers a GUI for interaction with human agents. The `start_vertex`,



(a) GUI of the reasoning agent

(b) internal structure

Figure 4.21: Reasoning agent

`target_vertex` and the `network` need to be specified. The results of dependency evaluation are displayed in a text area within the agent GUI. The reasoning agent also exchanges KQML messages with other agents (e.g., design agent). In the present implementation, however, only selected performatives such as *tell* and *ask* are used.

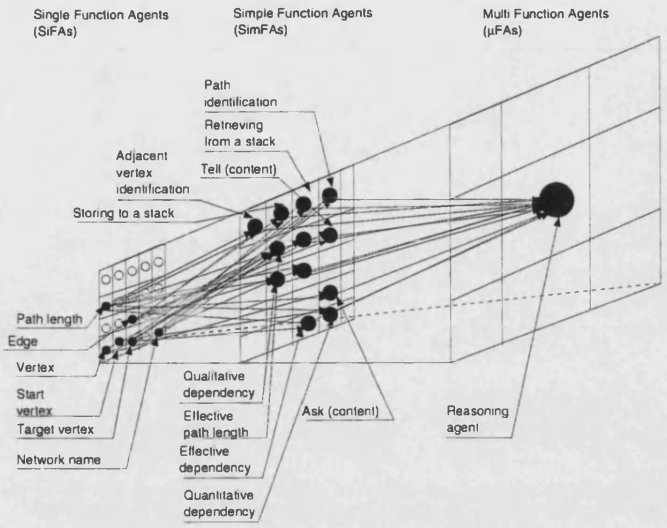
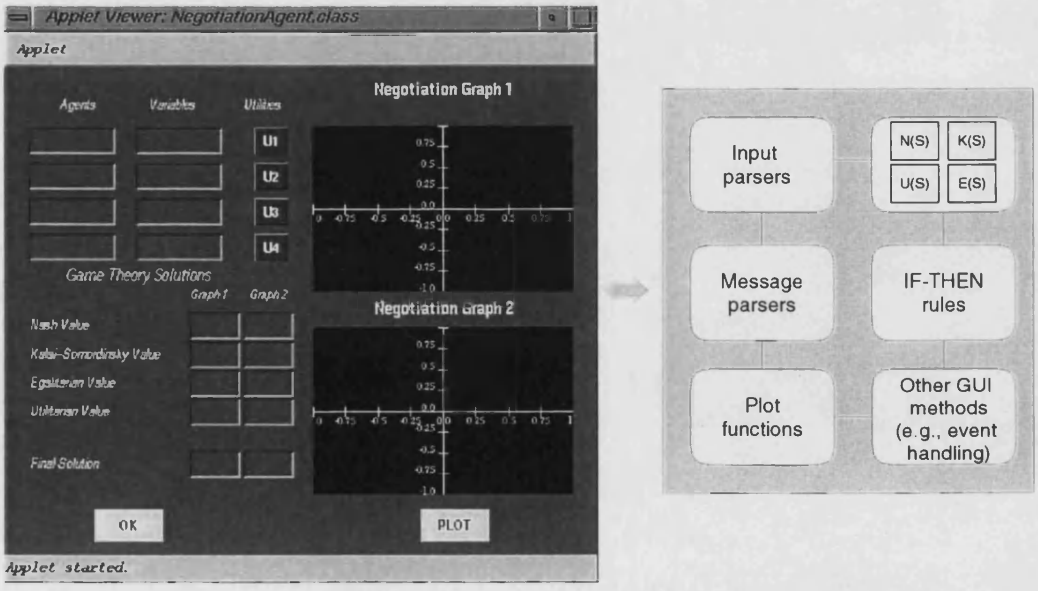


Figure 4.22: Transition from SiFAs to a μFA in reasoning agent

4.4.2 Negotiation agent

A set of four game theory techniques and rules provide the support for the negotiation agent [Figure 4.23]. The negotiation agent developed here is designed to solve two-agent conflicts between certain specified utility descriptions. The utility functions are specified to the negotiation agent by the conflicting agents.



(a) GUI of the negotiation agent

(b) internal structure

Figure 4.23: Negotiation agent

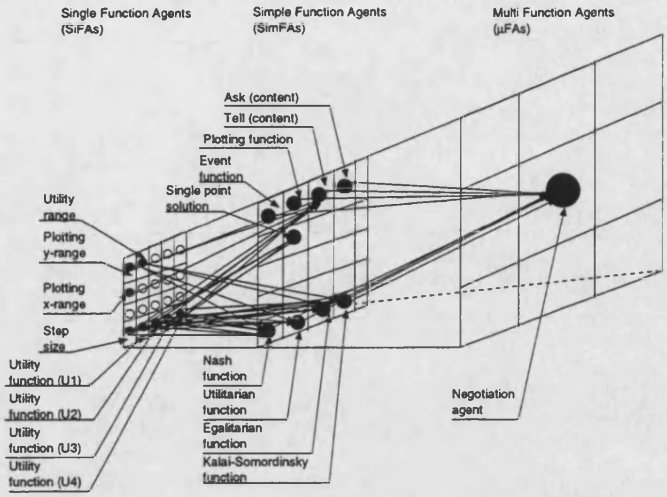


Figure 4.24: Transition from SiFAs to a μFA in negotiation agent

Figure 4.24 shows the formation of the negotiation agent by means of a transition from SiFAs to a μFA. The negotiation agent which is a μFA consists of parsers for KQML messages and utility functions [shown as Simple Function Agents (SimFAs) in Figure 4.24]. The KQML message parser consists of a set of methods to process performatives such as *tell* or *ask* which are processed as ASCII string objects. The negotiation agent also provides a plot function that plots the joint utility space (also called a biloss map). The final negotiated solution is evaluated and the outcome is sent back to the conflicting agents. Based on the nature of the obtained solutions, the agent uses additional decision enforcement rules to finalise a settlement. In the current implementation, the negotiation is designed to handle conflict resolution between two agents.

4.4.3 Implementation of reasoning and negotiation agents

The reasoning and negotiation agents were developed using Java programming language. The Java Development Toolkit-1.2 (JDK1.2) was used to develop the GUI components for the agents.

The input for the reasoning agent was specified in a file of which each line corresponded to an edge of the network graph. As each line of the input file is read, the reference to vertices and the dependencies are stored. The data structures associated with storage and retrieval of vertices (e.g., hash tables, stacks, linked lists, etc.) in a graph were adapted and modified from Weiss (1998). The methods for parsing the input from the text fields in the agent GUI [Figure 4.21], procedure for network reduction in Section 4.3.2 were implemented. This procedure for network reduction is a modification of the depth first search. Once the start and the target vertices are specified, the network reduction procedure is used to evaluated the effective dependencies.

The negotiation agent was designed specifically for two agent games. The agent utility functions for the conflict variables ($U1, U2, U3$ and $U4$) are read from a file. The four different game theory methods for the evaluation of negotiated solutions were coded a priori. The rules for the isolation of a single solution [Section 4.3.4] were also implemented. In addition, the methods for plotting graphs ["Negotiation Graph 1 and Negotiation Graph 2" in Figure 4.23], displaying results in the GUI, were also incorporated in the agent user interface.

4.5 Summary

This chapter presented the development of negotiation schemes for conflict resolution in an agent-based decision process. Game theory and dependency based reasoning have been first individually considered and illustrated through simple

examples of engineering design. A conflict resolution scheme has been proposed which has identified the positive attributes of the reasoning and negotiation techniques. The implementation of the new scheme has been presented with a view to utilise it in the agent-based product development process. The validation of the unified scheme in a collaborative process is presented in Chapters 5 & 6.

Chapter 5

Comparative case study: Collaborative design of a poppet relief valve

5.1 Introduction

In collaborative product development, software agents focusing on different aspects of a design have individual preferences over certain shared attributes of the product. This can result into different values for the shared design variables, thus leading to conflicts between agents. A novel scheme for conflict resolution in a decision process was developed in Chapter 4. There is a need, however, to apply and evaluate the new scheme in the context of a collaborative design process. To achieve this, a case study was carried out on the design of a poppet relief valve, aimed at comparing the new conflict resolution scheme with the previous work.

5.2 Objective

The objective of this case study was to test the feasibility of the negotiation scheme proposed in Chapter 4. The case study involved the design of a poppet relief valve which enabled the comparison with previous schemes for conflict negotiation[1].

In aerospace engineering and other industrial applications, valves are commonly used to regulate fluid flow and to relieve excessive pressures in closed conduits. Figure 5.1 shows a poppet relief valve consisting of helical spring, valve stem, and pipe enclosure. The helical spring and valve stem are enclosed in the pipe. The fluid is allowed to flow through the valve from inlet to outlet. The design principle here is that fluid flow occurs only when the pressure of fluid exceeds the cracking pressure. The fluid flow is cut off for fluid pressures below the cracking pressure where the helical spring presses the seal against the valve inlet thereby preventing any fluid flow.

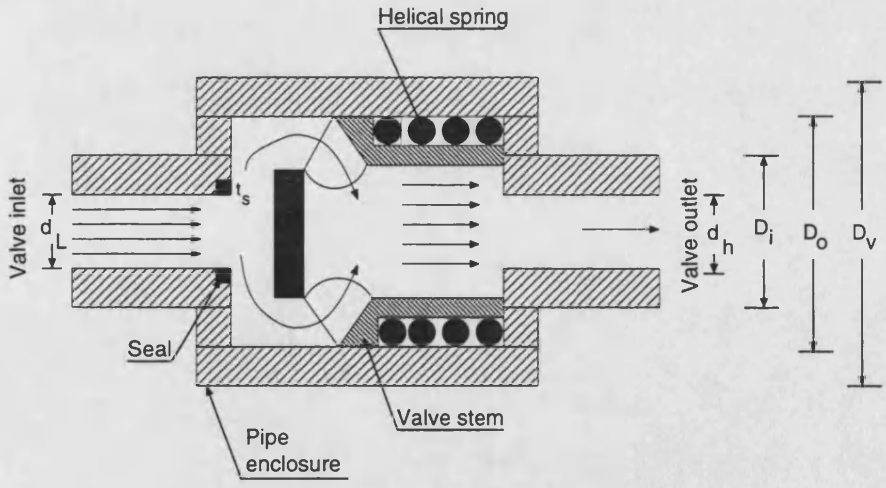


Figure 5.1: Schematic of a poppet valve

[1]This example served as a useful benchmark for negotiation since it was previously studied in a similar context by Kanappan and Marshek (1993) and Kusiak, *et al.*, (1996). The analytical formulae and design specification were taken from Lyons (1982) and Kanappan and Marshek (1993) and are restated in Appendix A.

5.3 A collaborative design process

In this study, the design of poppet relief valve is viewed as a collaborative process carried out by several software agents [Figure 5.2]. Each agent is associated with a particular perspective, viz, valve design, spring design and pipe enclosure design. Given a specification, the agents carry out the design task in collaboration. Details of the design problem are given in Appendix A.

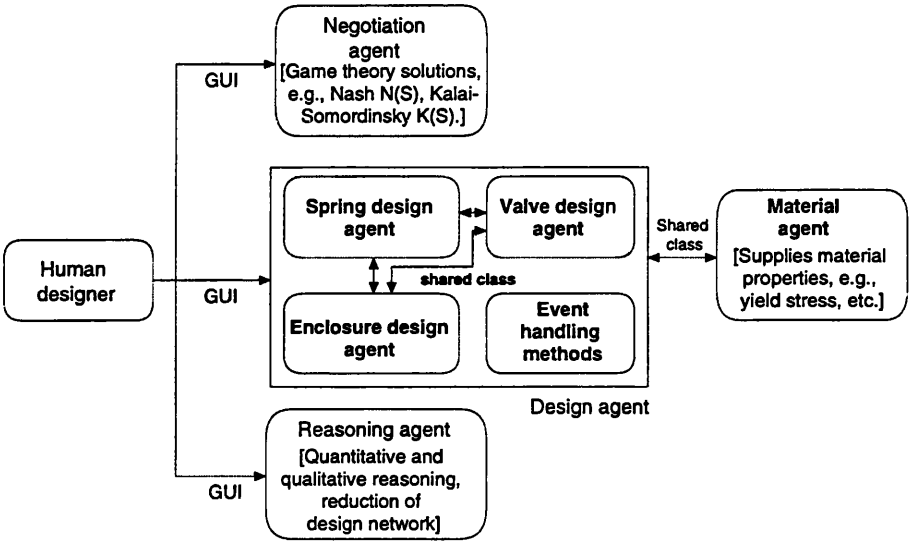


Figure 5.2: Design configuration of the poppet relief valve

The user inputs the design specification to the valve design agent which then evaluates its associated variables such as the valve outer diameter (D_v). The design specification and the shared variables are accessible to other agents via a shared class mechanism. Once the evaluation of variables by the valve design agent is complete, the spring- and the enclosure agents compute their associated variables (e.g, orifice diameter, in-line diameter, etc.). A difference in the values of shared variables (the internal and external diameters, D_i and D_o) leads to conflict between the agents. As a part of the conflict negotiation process, the human designer specifies certain preferences for the conflicting variables with respect to the perspectives of the agents involved in conflict. These preferences are represented in the form of utility functions and are input to the negotiation agent which evaluates a negotiated settlement. Figure 5.3 shows an overview of

the collaboration process used in this case study.

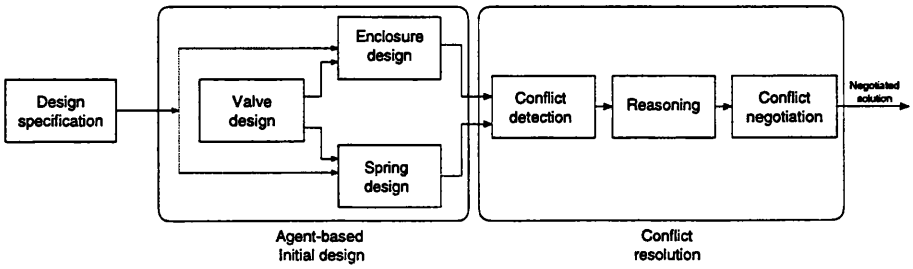


Figure 5.3: An overview of the collaborative process in the case study on poppet relief valve

5.3.1 Design specification

The designer specified the overall design requirements and constraints to the design agent as shown in Table 5.1.

Table 5.1: Design specification for poppet relief valve case study

Overall specification			
Fluid flow rate,	Q	m^3/s	25.236
Cracking pressure,	P_c	Pa	6.894×10^4
Max. fluid pressure,	P	Pa	1.723×10^6
Pressure drop,	δP	Pa	1.034×10^5
Fluid specific gravity,	S	-	1.0
Other initial variables			
Valve		Spring	Enclosure
$r_h = 1.0$		$rc_1 = 0.1$	$A_1 = 0.0005$
$C_f = 0.65$		$rc_2 = 0.1$	$A_2 = 0.0005$
$C_v = 1.3$		$N = 2.5$	$D_v = 0.0889$
$t_s = 0.00254$		$A_{cl} = 0.1$	-
$r_1 = 1.0$		-	-
$r_2 = 0.9$		-	-
Constraints			
$0.0216 \leq L_i \leq 0.0254, \quad 7 \leq C \leq 13$			

5.3.2 Initial design and conflict detection

The valve and spring agents evaluated the unknown variables and formed the initial design. A comparison of common variables indicated the presence of conflicts between the spring and enclosure agents on the values of internal and external diameters (D_i and D_o). These were determined as: $(D_i^s, D_o^s) = (0.0442m, 0.0876m)$

by the spring agent and $(D_i^e, D_o^e) = (0.0652m, 0.0896m)$ by the enclosure agent. Thus a conflict was detected between the two agents on the values of D_i and D_o.

5.3.3 Reasoning

The first step in conflict resolution was the dependency-based reasoning on a network of design constraints. The designer specified the constraint network of design variables to the reasoning agent. The reasoning agent determined the effective dependencies between the decision and performance variables, as shown in Table 5.2.

Table 5.2: Evaluation of dependencies between decision- and performance variables

Variables	Qualitative dependency, δ	Quantitative dependency (ψ) Kusiak, et al., (1996)	Quantitative dependency, (ψ) Present research
$D \rightarrow D_o$	+	0.9558	0.9519
$D \rightarrow D_i$	+	1.0588	1.0657
$d \rightarrow D_o$	+	0.0435	0.0480
$d \rightarrow D_i$	-	0.0588	-0.0657
$rc_1 \rightarrow D_o$	+	0.0862	0.0865
$rc_2 \rightarrow D_i$	-	0.1176	-0.1184
$A_1 \rightarrow D_o$	-	0.0102	0.0057
$A_2 \rightarrow D_i$	+	0.0113	0.0114
$D_v \rightarrow D_o$	+	1.0102	1.0057
$d \rightarrow V_s$	+	0.0674	-
$D \rightarrow V_s$	-	1.4869	1.4868
$D_v \rightarrow V_v$	+	0.0210	-
$C \rightarrow \tau_s$	-	-	-0.0600

The values for various dependencies compare well with those reported by Kusiak, *et al.*, (1996)[2]. These dependencies between the decision and performance variables are used subsequently in the negotiation stage which is discussed next.

5.3.4 Negotiation

The negotiation between conflicting agents was carried out by the negotiation agent based on the game-theoretic approach of Chapter 4. This involved the following three steps:

- Formation of the utility functions for *controlling decision variables* with respect to each conflicting agent perspective.
- Evaluation of the utilities for the *conflicting variable* with respect to each conflicting perspective.
- Evaluation of alternative *negotiated solutions.*

(a) Utilities for controlling decision variables

For the spring design agent, the spring index, C, was chosen as the controlling decision variable. This choice was based on the constraint on the installed length of the spring, L_i.

For the enclosure design agent, the corrosion resistance allowance A_2 was selected as the controlling decision variable. This was the only design variable that could be relaxed to generate utility for D_i without violating the design specification. The utility functions specified for the respective controlling variables

[2]The minor differences can be explained by the fact that the values of constants rc_1 and rc_2 used by Kusiak were not reported.

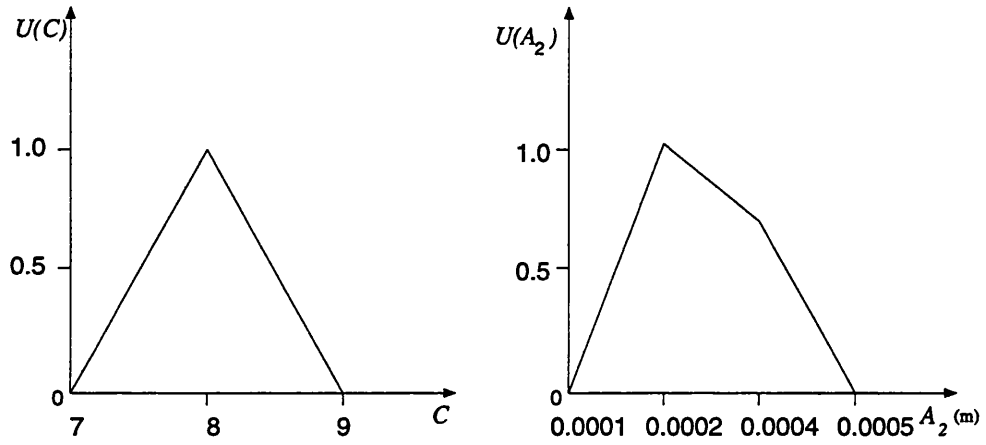


Figure 5.4: Specified utility functions for controlling decision variables

for the spring and enclosure agents are shown in Figure 5.4. The profile for C in Figure 5.4(a) is based on the fact that for the limiting constraint on $C = 7$, $U(C)$ is zero. The utility for spring agent increases until $C = 8$ where it is maximum and then shows a decrease. The optimal design is bounded by the constraints on C. As for corrosion resistance allowance (A_2), the utility is zero at $A_2 = 0.0001m$. Based on experimental results reported by Lyons (1982), the most preferred value lies in the range $0.00015m - 0.0002m$. The allowance under the given specifications A_2 should not exceed the maximum allowed value of $0.0005m$.

(b) Utilities for conflicting variables

The utilities for the conflicting variables D_i and D_o were evaluated form the dependencies in Table 5.2 and the specified utilities in Figure 5.4. These are shown in Figure 5.5. Figures 5.5 (a) and (b) show the evaluated utilities for D_i and D_o for the spring design agent. The utilities for D_i and D_o for the enclosure design agent are shown in Figure 5.5 (c) and (d). Figure 5.5 (a) shows that for the spring agent any value for D_i in the range (0.0449 - 0.0655 m) is acceptable with 0.0548 m being the most preferred value. On the other hand for

the enclosure agent [Figure 5.5 (c)] any value for D_i in the range (0.0439 - 0.0449 m) is acceptable with 0.0441 m being the most preferred value.

Since the negotiation on a conflicting variable would require an overlap of the acceptable ranges, these were plotted for each conflicting variable, D_i and D_o, as shown in Figure 5.5 (e) and (f). The overlapping region of the utilities represents the negotiation range for each variable. The overlapping utilities for D_o in Figure 5.5 (f) show a non-empty negotiation range suggesting the existence of a game-theoretic solution. However, for D_i, the negotiation range was null as shown in Figure 5.5 (e).

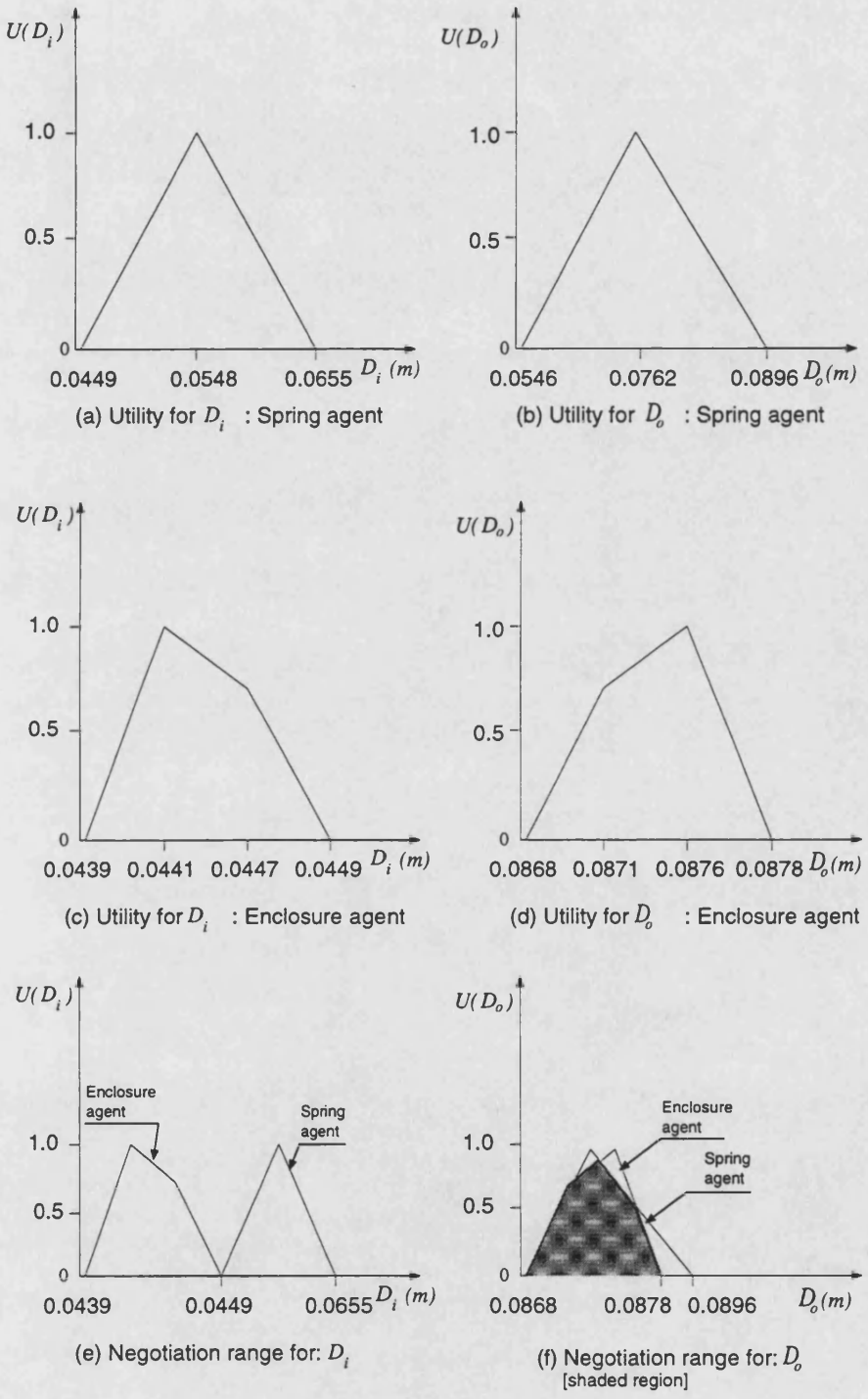


Figure 5.5: Evaluated utility functions for conflicting variables

(c) Redefinition of utilities for controlling design variables

Due to the empty negotiation range for D_i, agents were required to relax their constraints and their respective utility levels. Due to the physical nature of the problem the constraints for enclosure design agent could not be relaxed. For the spring design agent, the constraint on spring index C was relaxed to a wider range ($4 \leq C \leq 20$) and the utility function was redefined. Additional constraints were introduced for rc_1 and rc_2 ($rc_1 \geq 0.05$m; $rc_2 \geq 0.05m$) to ensure a negotiated range for D_i. The utilities for C, rc_1 and rc_2 are shown in Figure 5.6.

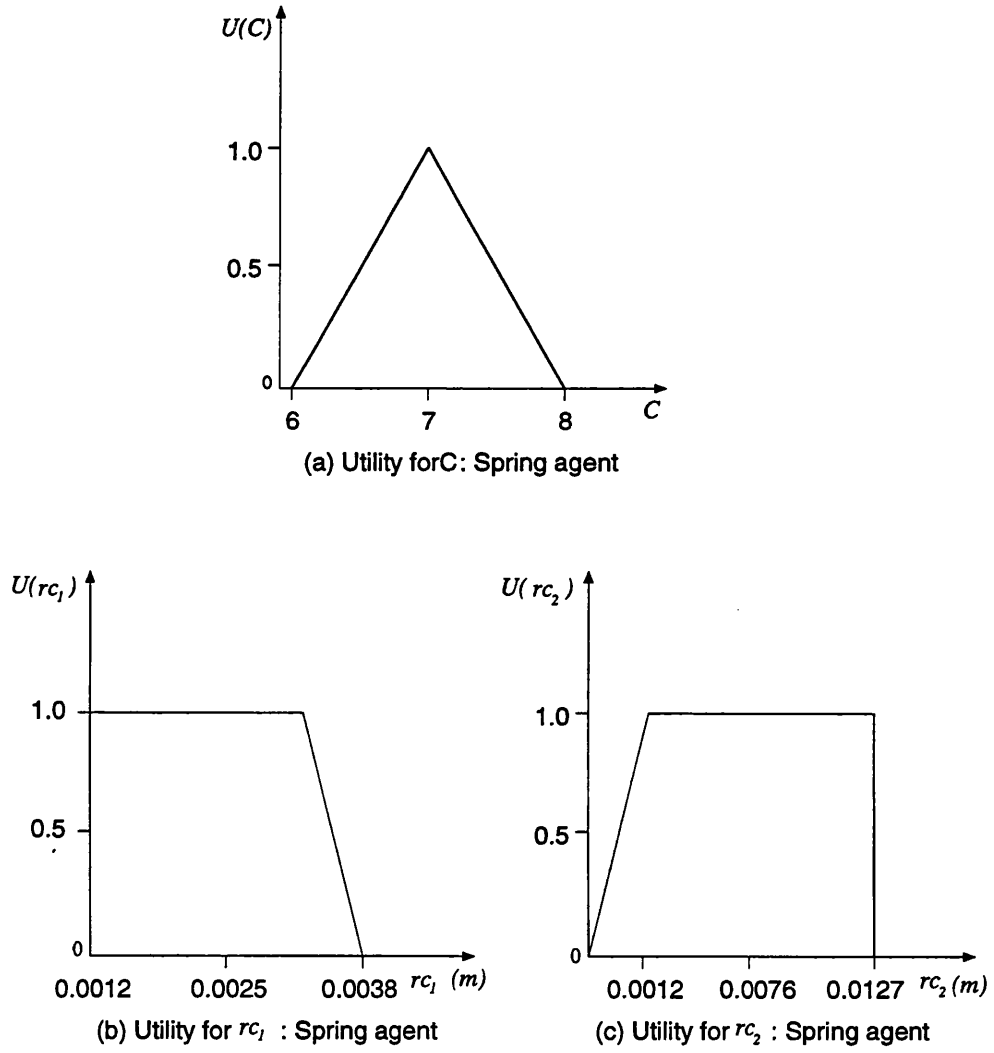
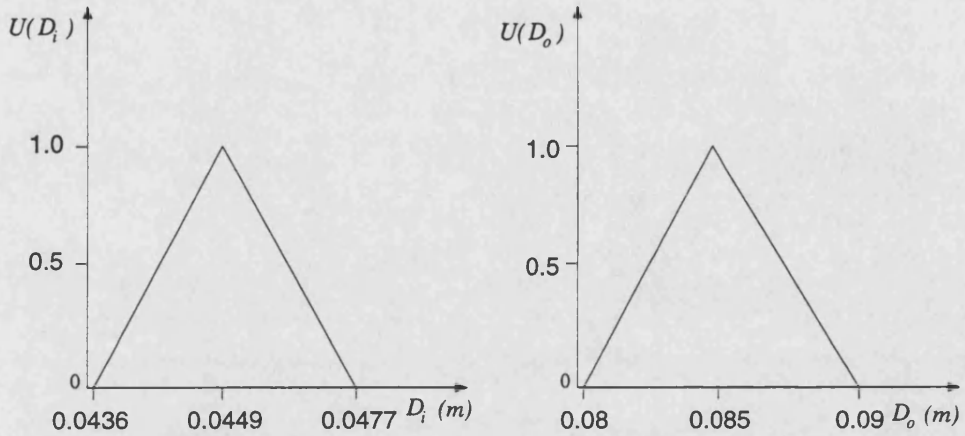


Figure 5.6: Redefined utility functions for the controlling decision variables for the spring agent

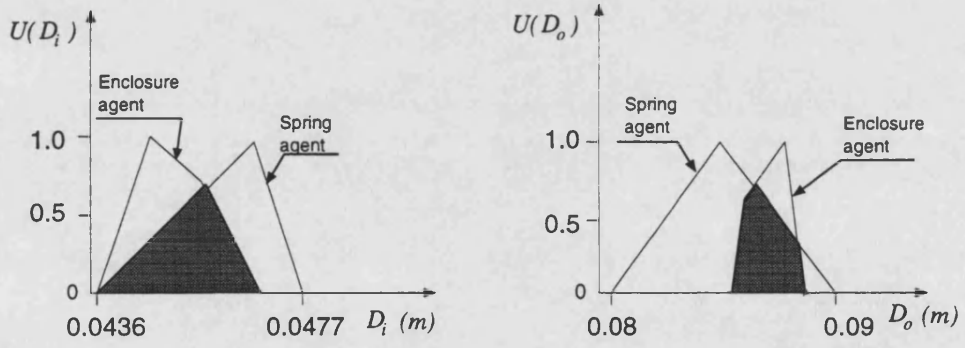
The re-evaluated utilities for D_i and D_o are shown in Figure 5.7. Figure 5.8

shows the negotiation range resulting from this constraint relaxation process, and confirms the existence of a negotiated solution for D_i and D_o.



(a) Utility for D_i : Spring agent (b) Utility for D_o : Spring agent

Figure 5.7: Re-evaluated utilities for conflicting variables for the spring agent



(a) Negotiation range for D_i [shaded region] (b) Negotiation range for D_o [shaded region]

Figure 5.8: Negotiation ranges for conflicting variables with re-evaluated utilities

(d) Negotiated solutions

The new utilities for D_i and D_o were input to the negotiation agent to determine a negotiated settlement. The negotiation agent plotted the joint utility outcome for each of the conflicting variables as shown in Figure 5.9. Four different game-theoretic methods were applied to obtain alternative negotiated solutions which are shown in Table 5.3. The corresponding utilities for the two agents are shown in Table 5.4. The Nash and Kalai- Somordinsky solutions compare well with those

obtained by Kanappan and Marshek (1993) who did not consider the Egalitarian and Utilitarian solutions. It can be observed that in this case the values of Kalai-Somordinsky and Egalitarian solutions turned out to be the same while the Nash and the Utilitarian solutions evaluated to mutually identical values. Due to the nature of Kalai-Somordinsky solution which ensured equal pay-offs for spring and enclosure agents, this solution was chosen as the final settlement. This final negotiated solution thus satisfied the design requirements and constraints while objectively meeting the goals and preferences of the agents.

Figure 5.9: Negotiation agent showing the utility curve for the conflicting variables

Table 5.3: Negotiated solutions

Conflict variables	Kanappan and Marshek (1993)		Present approach				Final solution
	N(S)	K(S)	N(S)	K(S)	E(S)	U(S)	
D_i, m	0.0441	0.0441	0.0447	0.0446	0.0446	0.0447	0.0446
D_o, m	0.0876	0.0876	0.0876	0.0871	0.0871	0.0876	0.0871

Table 5.4: Utility outcomes for the spring and enclosure agents

Joint utilities for conflict variables	Spring design agent				Enclosure design agent			
	N(S)	K(S)	E(S)	U(S)	N(S)	K(S)	E(S)	U(S)
$U(D_i)$	0.8	0.742	0.742	0.8	0.699	0.742	0.742	0.699
$U(D_o)$	0.571	0.671	0.671	0.571	1.0	0.671	0.671	1.0

5.4 Discussion

(a) Comparison of author's results with those of Kusiak, *et al.*,(1996):

In Kusiak's approach a perspective such as design was assumed to have several sub-problems. Each individual sub-problem was aware of the other sub-problems in the perspective and was willing to fully co-operate. Thus information transparency between sub-problems was allowed from the start of the design process to the end. Although termed as *negotiation*, this approach is not too dissimilar to optimisation by iterative search. This process differs from the negotiation process considered in this research in important ways.

- Each conflicting agent (representing a sub-problem) has its own priorities and generates its own utility functions independent of the other agents.
- An agent may choose to mis-represent its information in order to maximise its gain.
- As a negotiation session begins, the agents may make use of threats which

are very common in bargaining problems. For example, the conflict scenario considered in the present study could also be viewed as a two agent game with threats. When the first attempt to make a negotiated settlement failed due to a lack of common utility space, both agents (spring and enclosure) were required to relax their constraints and redefine their utility functions. For the enclosure agent, the value of corrosion resistance allowance for the poppet valve stem (A_2) could only vary between 0.00025m and 0.0012m under the given circumstances. Though the aspect of threats was not addressed in this case study, such a behaviour can be incorporated if necessary. Such a possibility was not studied by Kusiak, *et al.*

- Another major difference is that in Kusiak's work, any solution within the feasible design space was considered to be acceptable (i.e., so long as design constraints were not violated). In practice, however, a designer often needs to select the best solution based on some objective criteria of its preferences. In the current approach therefore the agents are able to state their individual preferences. Thus evaluating a feasible region is not the final settlement of conflicts. Conflicts often arises due to difference in preferences, however, Kusiak *et al.*, defined conflicts due to violation of design constraints.

- The present research differs in another aspect that single point solutions are obtained as a mark of reaching a settlement. In Kusiak's approach one cannot predict a priori the number of iterations needed to make a compromise.

(b) Comparison of author's results with those of Kanappan and Marshek (1993):

- The key difference between Kanappan and Marshek's game theoretic approach, and the present research is the use of dependency based constraint relaxation in conjunction with game-theoretic approach. This is crucial in engineering problems as we demonstrated in the initial design iteration

where the feasible design space for the agents did not intersect to form a non-empty negotiation space.

- Kanappan and Marshek's approach was based on static utilities. In the present approach we demonstrated the agents may need to change utilities similar to the aspiration based approach of Kersten (1989). This truly depicts a conflict situation where the agents may dynamically change their preferences during the negotiation.

- Kanappan and Marshek assumed only one controlling decision variable for each agent thereby restricting the choice of utility function for a given problem. The present study used as many as three controlling decision variables for the spring design agent. This was found to increase the flexibility in selecting a range of values for conflicting variables.

- Kanappan and Marshek did not present the solutions from the viewpoint of design consistency. In the present case, the consistency is ensured by means of qualitative and quantitative reasoning in a network of design variables.

- The present approach included selection strategies to isolate a negotiated solution from a set of four game-theoretic solutions by the use of IF-THEN rules.

- The inclusion of four methods in the negotiation scheme provided the designer with a greater choice for evaluating negotiated solutions. The Egalitarian and Utilitarian methods were not studied in this context by Kanappan and Marshek.

5.5 Summary

Conflict negotiation in engineering design is a frequent though difficult problem. A case study on conflicts in the design of a poppet relief valve was presented.

A new negotiation scheme was applied which examined conflict resolution between agents in a design process. This scheme made use of a combination of dependency-based reasoning coupled with game-theoretic methods. The reasoning ensured the formation of feasible design space whereas the game-theoretic methods isolated four single point solutions within this space. Rules for isolating a single solution were successfully tested for isolating a single point solution. The numerical results thus obtained were compared with the previous work and showed a good comparison with that of Kanappan and Marshek (1993). The results indicate the merits of the proposed negotiation scheme, such as preserving agents' preferences and maintaining rationality, over iterative negotiation models of Kusiak and Wang (1996). This scheme has been found to formally capture the combined benefits of game-theoretic and constraint based techniques applied to a two-agent negotiation problem.

Chapter 6

Validation case study: Collaborative design and manufacture of a Geneva mechanism

6.1 Introduction

This chapter is aimed at validating the agent synthesis and goal alignment hypotheses presented in Chapter 1 for an agent-based collaborative design process. The synthesis of agents including the modelling of Multi-Agents using the SiFA approach was presented in Chapter 3 and illustrated through the example of the Geneva mechanism. The problem of conflicts between the agents involved in collaborative design process was examined in Chapter 4 and a novel scheme was presented for conflict resolution by the alignment of goals. Chapter 5 showed the application of this scheme through the design case study of a poppet relief valve.

Having tested the two hypotheses and related methodologies in isolation, now there is a need to apply and test them together in a common and elaborate setting of laboratory based collaborative product development, involving both design and manufacturing. To achieve this, a case study was carried out on collaborative design and manufacture of a Geneva mechanism.

6.2 Objective

The objectives of this case study were as follows:

- To validate the new Single Function Agent (SiFA) based agent synthesis approach;
- To validate the new conflict resolution scheme in a collaborative product development process, including manufacturing and consisting of both software agents and human decision makers.

The Geneva mechanism was chosen as the test vehicle for this case study for the following reasons:

- It is an assembly rather than a single component;
- It has reasonably complex design function and associated functional behaviour. Therefore the associated design agents would also be more complex;
- It represents a product of higher complexity with respect to its geometry;
  - Large number of design variables
  - Tightly coupled design variables

  - Highly interactive geometrical features
  - Tighter tolerances

- Its manufacturing process[1] would be complex;
  - Need for high quality precision tools due to tight tolerances.
  - Additional consideration of shrinkage.
  - Design of ejection mechanism.

Whereas the design of a poppet relief valve applied the conflict resolution scheme in Chapter 4 to software-to-software negotiation mode only, this case study tested the conflict resolution scheme in three distinct negotiation modes:

- software agent - software agent (type 1 conflict);
- software agent - human agent (type 2 conflict);
- human agent - human agent (type 3 conflict).

Therefore, this case study aims to enhance the understanding developed in Chapter 5 and tests the scalability of this thesis by addressing the following:

- Increasing the complexity of the product;
- Broadening of the scope of product development process;
- Broadening of scope for conflict negotiation.

[1]Geneva mechanisms are usually manufactured using metals (typically steel or aluminium). To suit one of the experimental objectives of including the stand-alone injection moulding machine, the present case study has considered the manufacture of the mechanism from polymeric materials.

6.3 Experimental methodology

The case study consisted of the following key phases apart from the additional tasks of conflict resolution:

- Organisation
- Construction of the problem domain
- Design for function
- Design for manufacture
- Development of manufacturing tool
- Product manufacturing

Further sections will address each of these phases.

6.4 Organisation

6.4.1 Team formation

This phase involved the identification of the following:

- Skills required
- Responsibilities of each team member
- Resources

The humans in this collaborative test-bed include individuals with relevant expertise in design, drafting, manufacturing and project management [Table 6.1]. The roles played by humans include manual as well as supervisory control of tasks. The manual control involves tasks such as mould making. Most of the manual tasks are carried out by HA3 and HA4. The designer (HA2) is responsible for mould design and operation of an injection moulding machine. HA1 is responsible for product design and creation of part drawings. Human agents HA3 and HA4 are responsible for mould and product manufacturing. The supervisory control (exercised by HA1 and HA2) involves tasks such as monitoring the machining operations, decision-making at various stages (e.g., positioning of the injection point on the mould, selection of plastic material), negotiation of conflicts and providing management support. The role of human agent HA5 is to advise in collaborative product development, provide management support and critique the results.

Table 6.1: Team members and their main roles

Members of the team	Level of expertise	Main responsibilities
HA1	Research student	- development of software agents - creation of part drawings
HA2	Research student	- Mould design - product manufacturing
HA3	Technician	- mould manufacturing
HA4	Technician	- mould manufacturing
HA5	Member of academic staff	- product inspection - management support

Most of the interaction in this phase between humans involved face-to-face communication to discuss the logistics of the experiment. The tools needed to carry out the experiment were identified. These tools included a platform for developing software agents, manufacturing machines and commercial off-the-shelf software such as WebSpaceAuthor for the visualisation of VRML models. A part of

the phase also involved the use of asynchronous tools such as e-mail in order to schedule meetings. The overall communication process itself was informal.

6.4.2 Configuration of the test-bed

The available facilities at the University of Bath that represented a balanced set of tools for collaborative work were identified. The main steps involved in preparing the test-bed were: the development of software agents and examine how agents are applied in a conflict situation.

Other commercial off-the-shelf tools that were newly identified and considered to be useful in collaborative were Solidworks for the conversion of AutoCAD drawings to VRML, WebSpace Author for the conversion of 3D VRML models to Inventor standards. In addition, several manufacturing facilities were also identified that could form a part of the collaborative process. Figure 6.1 shows the software tools and facilities selected for the present case study. The tools were configured into a test-bed for collaborative product development as shown in Figure 6.2.

6.4.3 The product development process

The product development process was designed and carried out to address all the tasks set for this case study, as outlined in section. Figure 6.3 shows an IDEF0 model of process and the information flow in the collaborative design and manufacture of the Geneva mechanism. The tasks in the collaborative process are jointly carried out by software agents and humans. The figure also shows the deployment of conflict resolution activities in the product development process. The following sections describe the individual activities in the product development process.

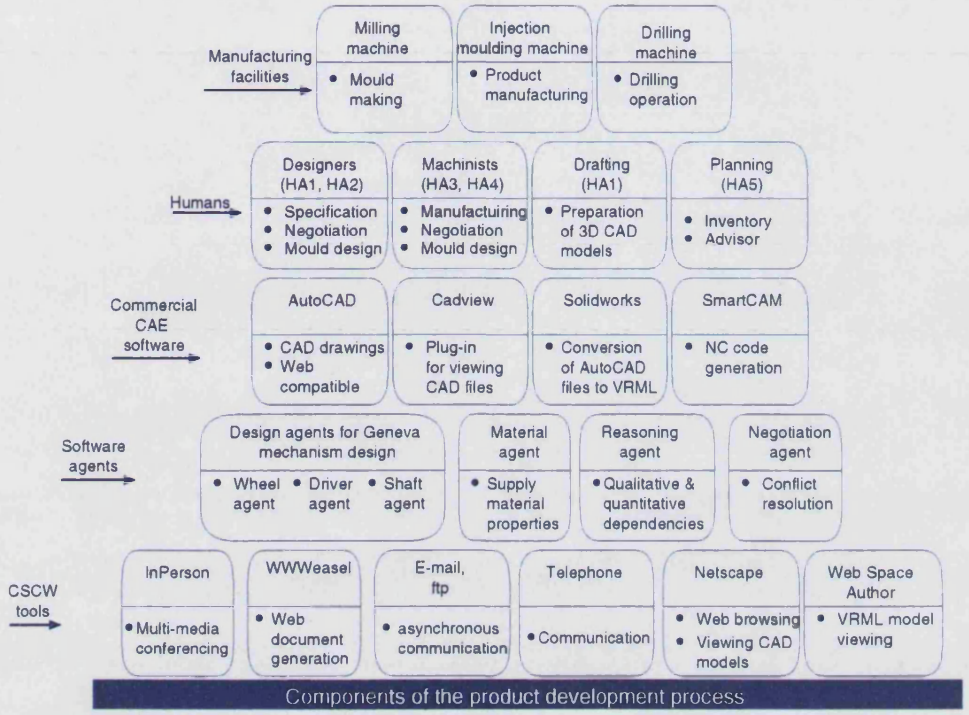


Figure 6.1: Components of the collaborative product development

6.5 Deployment of agents in the collaborative process

The software agents for this case study were developed using the Java programming language, as Java is especially suited for networked computing. The agents thus developed were generic as well as problem specific. The collaborative process shown in Figure 6.3 consisted of three generic agents. These are the material agent [Section 3.4.3], the reasoning agent and the negotiation agent [Sections 4.4.1 and 4.4.2]. The problem-specific agent was itself a Multi-Agent, which was developed for the design of the Geneva mechanism [Section 3.4.4]. The deployment of these agents in a collaborative design process with reference to conflict resolution is discussed in this section. The collaborative activities can be divided into four stages: pre-conflict, reasoning, conflict resolution and post-conflict.

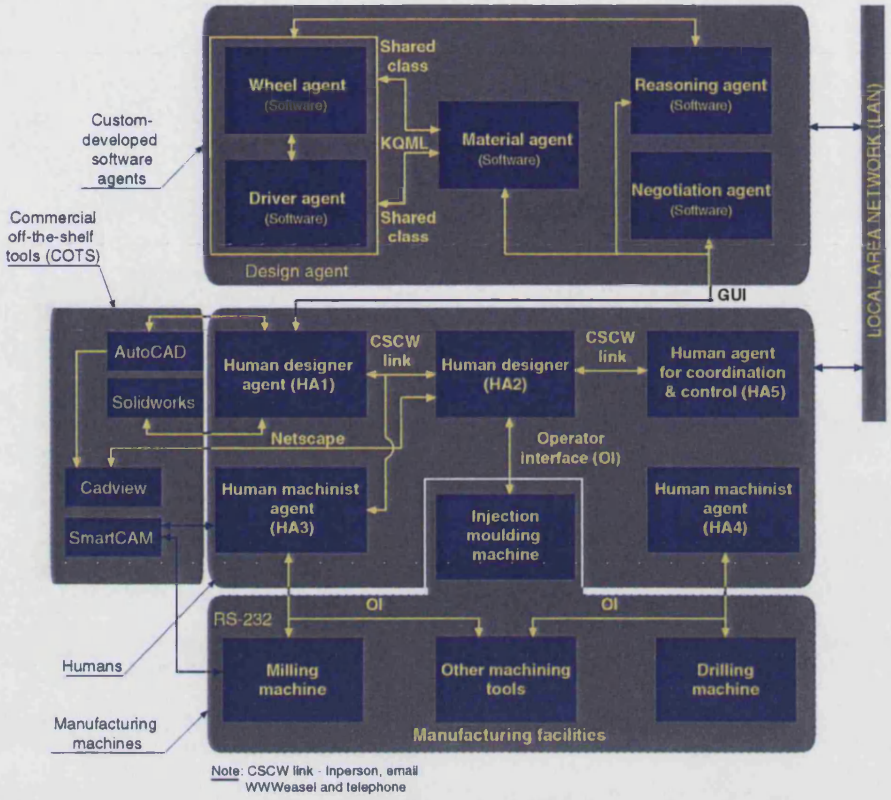


Figure 6.2: Configuration of product development tools

6.5.1 Pre-conflict stage

This stage involves the events leading up to a conflict. The problem-specific design agent is deployed at this stage for the design of the product, ie. the Geneva mechanism. The design agent here consists of two μFAs for wheel and driver design [Section 3.3.4]. The key steps involved in the pre-conflict stage are the following:

[a1.] The domain equations relating to the wheel and driver kinematics are pre-coded in their respective μFAs.

[a2.] Each μFA has its own performance variables pre-specified: pin contact stress (τ_C) for the driver and root stress (σ_R) for the wheel.

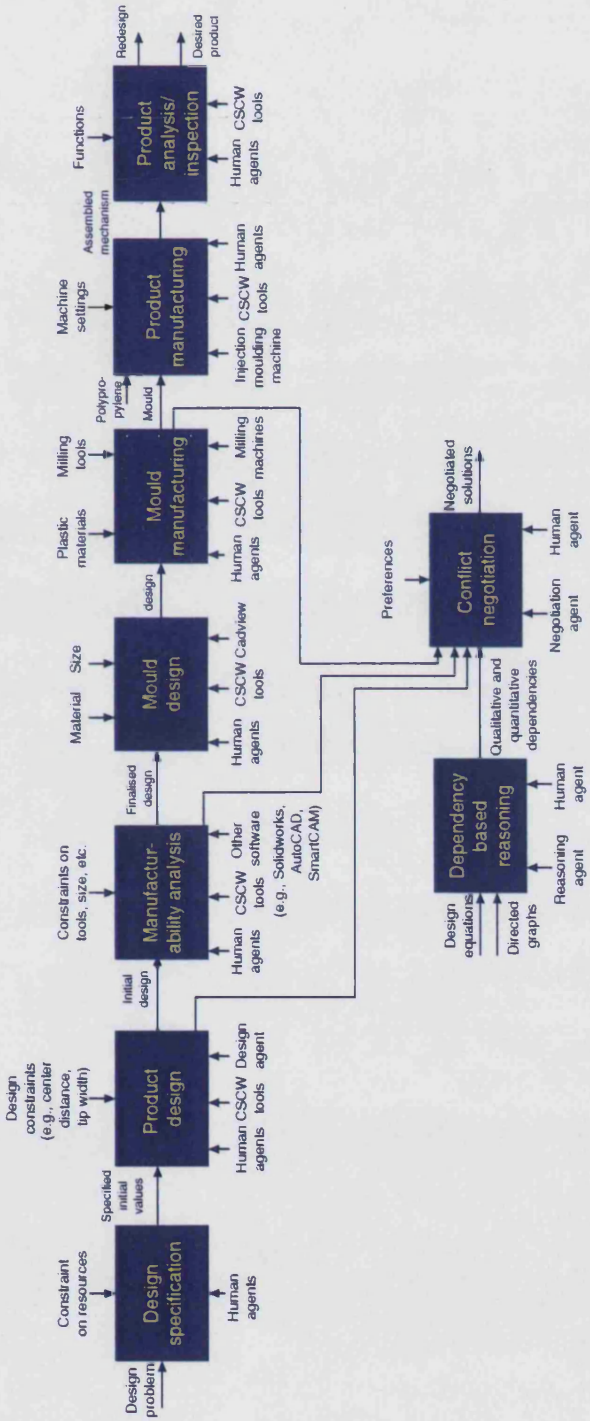


Figure 6.3: IDEF0 representation of the product development process

[a3.] The controlling decision variables for each μFA are also pre-specified. This choice is determined by the domain equations.

[a4.] The design requirements are input to the design agent by the designer. The material agent supplies the required material properties to the design agent.

[a5.] The designer specifies the primary variable for variational design, its range and the required number of design variations to be generated.

[a6.] The design agent determines an initial design from the perspective of each μFA. The initial design would contain inconsistencies due to the conflict between perspectives.

[a7.] The design agent identifies the shared variables in conflict.

[a8.] A number of variational designs of the initial design in [a6] are generated as specified in [a5].

6.5.2 Reasoning stage

Upon the detection of a conflict, the design agent initiates the reasoning stage by invoking the reasoning agent. This stage involves the determination of effective dependencies between the conflict variables and the controlling decision variables for each μFA. The following steps are carried out to enable agent based reasoning during the collaborative process:

[b1.] The designer specifies the nodes and edges of the design network graph to the reasoning agent (problem-specific information).

[b2.] The reasoning agent forms the internal graph representation.

[b3.] The designer specifies the controlling decision variable (start vertex) and the conflict variable (target vertex).

[b4.] The reasoning agent identifies all paths between the decision and conflict variables.

[b5.] The reasoning agent obtains all edge dependencies from the problem-specific design agent (automatically determined by the design agent).

[b5.] The reasoning agent evaluates the effective dependencies between the decision and conflict variables.

Steps [b3] - [b6] are also repeated to all pairs of decision and performance variables.

The effective dependencies are stored for further processing.

6.5.3 Conflict resolution stage

The designer invokes the negotiation agent by invoking the negotiation agent. The following steps are involved:

[c1.] The designer specifies the utility functions for the controlling decision variables for each μFA (from [a3]) over the range of variational designs obtained in [a8].

[c2.] Each μFA (eg., wheel and driver agents) constructs its own utility function(s) for the conflict variable(s) by using the information from [c1] and [b8].

[c3.] The design agent identifies the negotiation range for each conflict variable using information from [c2].

[c4.] If the negotiation range is not null, the design agent sends the utilities from [c2] to the negotiation agent. Otherwise it asks the designer to modify the utilities in [c1].

[c5.] The negotiation agent plots the joint utility between the agents for each conflicting variable.

[c6.] The negotiation agent determines the game theory solutions based on the four methods discussed in Chapter 4.

The negotiation agent isolates the final negotiated solution based on the equal pay-off criteria.

6.5.4 Post-conflict stage

This stage incorporates the results of negotiated settlement into rest of the design. It consists of the following key step:

[d1.] The negotiated value for the conflict variable is used to generate the final design. This is carried out by the reasoning agent using the steps in [b8]. The changes can be local or global depending on the problem at hand and are generally not known a priori.

6.6 Construction of the problem domain

6.6.1 Problem definition

Geneva mechanisms are used widely as indexing mechanisms in machines (e.g., shaping tools, conveyers, feeding devices and other elements that have intermittent motion). A simple Geneva mechanism is shown in Figure 6.4. It consists of a slotted wheel, a driver with a driving pin and a supporting or mounting base.

Various geometrical variables involved in the design of the Geneva mechanism and related mathematical equations are given in Appendix C.

Generally, the design of the Geneva mechanism is carried out in the following two stages:

1. **Mechanism selection:** First the kinematic inputs are specified by describing the desired input and output motion characteristics (motion profile or velocity and acceleration curves) and dwell time [Yeaple (1979)]. The designer then selects a mechanism with key geometric parameters based on data charts of motion characteristics [Hasty and Potts (1966)].

2. **Detailed parametric design:** Given the basic geometry, this stage involves the investigation of undesirable motion characteristics such as high contact stress between drive pin and wheel slot, and vibratory motions the mechanism. The design specifies the required number of slots (N_s), inertial load (I_l) and rotational speed of the driver (ω), and the problem is to evaluate the design characteristics mentioned above [Johnson (1956), Lee (1981)].

For the purpose of this case study, only parametric design of stage 2 was considered. The design and manufacture of the Geneva mechanism was therefore to be carried out by specifying N_s, ω, I_l, P_{mat} and additional associated constraints. The mechanism was to be manufactured using a grade of polypropylene.

6.6.2 Specification

The designer HA1 was responsible for inputting the specification for the Geneva mechanism to the design agent. The design of the Geneva mechanism was specified as shown in Table 6.2. This information was shared with other agents using the shared class mechanism.

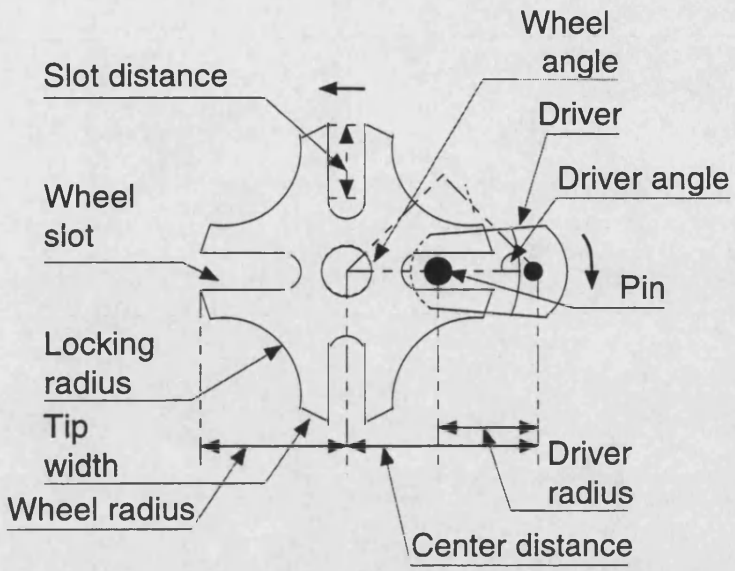


Figure 6.4: Schematic of a four-slot external Geneva mechanism

Table 6.2: Design specification and constraints for the Geneva mechanism

SPECIFICATIONS	
Number of slots, N_s	6
Speed of rotation, ω (rpm)	1000
Inertial load, I_l (Kg m^2)	1.1296 x 10^6
Material, P_{mat}	Polypropylene
CONSTRAINTS	
$0.003 \geq w_{tip} \leq 0.007$;	$w_t \leq 0.01m$
$0.05 \leq D \leq 0.07$;	$N_s \leq 8$
$C \leq 0.03m$;	

6.6.3 Evaluation of dependencies

Following the specification, the next task was for the reasoning agent to evaluate the dependencies between design variables using the domain equations given in Appendix C. Based on the specification and domain equations, the reasoning agent evaluated the influence relationship for each edge as shown in Table 6.3. Table shows the dependencies evaluated for a Geneva mechanism made from polymeric materials. For the purpose of demonstration, the dependencies were also evaluated for plain carbon steel and are shown in the table where even the

choice of material has an influence. The dependency graph for the mechanism was constructed by the reasoning agent in a computational form by using the data in Table 6.3, read from a file. This was done in order to enable the reasoning agent to carry out network reduction.

6.6.4 Network reduction

The purpose of network reduction is:

- To evaluate the effective dependencies between decision and performance variables;
- To identify the best candidates for negotiation variables, based on sensitivity analysis.

Having specified the network topology, the next step was to evaluate the effective dependencies between the decision- and performance variables. The key decision variables were C, D, w_{tip}, r_p and S_w and the key performance variables were τ_C, σ_R and σ_T. Among the decision variables, slot width, S_w, was particularly important as it was a common variable for both driver and wheel agents. Therefore this was a candidate to be used as *controlling* decision variable during negotiation.

The effective qualitative and quantitative dependencies between decision and performance variables were determined as shown in Table 6.4 and the corresponding reduced network is shown in Figure 6.6. This required the identification of the number of paths (NP) and then apply the reasoning procedure based on serial and parallel dependencies presented in Chapter 4.

Among the three performance variables shown in Table 6.4, the contact shear stress (τ_C) between the pin and the wheel slot is of the greatest importance

Table 6.3: Qualitative and quantitative dependencies between the design variables for the Geneva mechanism corresponding to the specifications (in Table 6.2)

Influence relationship (directed edge)	Qualitative dependency (δ)	Quantitative dependency (ψ)	Variables	Qualitative dependency (δ)	Quantitative dependency (ψ)
$\beta_0 \rightarrow r_d$	+	0.89278	$X \rightarrow Y$	+	0.39849
$\beta_0 \rightarrow S$	-	-0.89278	$X \rightarrow \sigma_T$	-	-0.67473
$\beta_0 \rightarrow h_t$	-	-1.08096	$Y \rightarrow h_t$	-	-1.11748
$\beta_0 \rightarrow h_R$	-	-5.30534	$r_L \rightarrow I_g$	-	PP -4.06264 Steel -4.10939
$\beta_0 \rightarrow \sigma_R$	+	0.00162	$K_1^s \rightarrow \sigma_R$	+	1.00514
$\beta_0 \rightarrow \alpha_0$	-	0.5	$K_2^s \rightarrow \sigma_R$	-	-0.00514
$D \rightarrow C$	+	1.0	$\mu \rightarrow \sigma_R$	+	2.19963
$D \rightarrow C_L$	+	1.0	$h_r \rightarrow \sigma_R$	-	-1.73978
$D \rightarrow h_t$	-	-1.02476	$h_t \rightarrow \sigma_t$	-	-2.99379
$D \rightarrow w_t$	+	1.0	$r_p \rightarrow \sigma_C$	-	-0.46537
$D \rightarrow \rho_{max}$	+	1.0	$r_p \rightarrow S_w$	+	0.89361
$S \rightarrow h_R$	-	-14.06	$I_g \rightarrow P_d$	+	PP 0.01067 Steel 0.19920
$S \rightarrow I_G$	-	PP -0.63497 Steel -0.64361	$I_l \rightarrow P_d$	-	PP -0.47578 Steel -0.66432
$P_d \rightarrow \sigma_C$	+	0.48808	$\rho \rightarrow I_g$	+	PP 0.99819 Steel 0.98480
$P_d \rightarrow \sigma_R$	-	-0.00514	$\rho_{max} \rightarrow P_d$	-	-0.14299
$P_d \rightarrow \sigma_T$	-	-0.67473	$N_s \rightarrow I_g$	+	PP 6.16095 Steel 6.22063
$w_t \rightarrow \sigma_C$	-	-0.46537	$\mu' \rightarrow \tau_c$	+	1.0
$w_t \rightarrow \sigma_R$	+	0.00467	$\mu'' \rightarrow P_d$	+	0.09784
$w_t \rightarrow \sigma_T$	-	-2.29316	$K \rightarrow \rho_{max}$	+	1.0
$C \rightarrow S$	+	1.0	$C_L \rightarrow S_w$	+	0.017819
$C \rightarrow r_d$	+	1.0	$w_{tip} \rightarrow r_L$	-	-5.93736
$C \rightarrow I_G$	+	PP 22.5232 Steel 22.9654	$w_t \rightarrow I_g$	+	1.0
$\omega \rightarrow a_{max}$	+	1.0	$a_{max} \rightarrow P_d$	+	0.23255
$\omega \rightarrow a_v$	+	1.0	$K_3 \rightarrow w_t$	+	1.0
$S_w \rightarrow h_t$	+	0.27257	$\sigma_c \rightarrow \tau_c$	+	1.0
$S_w \rightarrow I_G$	+	PP 0.81824 Steel 0.82936			
$S_w \rightarrow P_d$	+	0.09784			

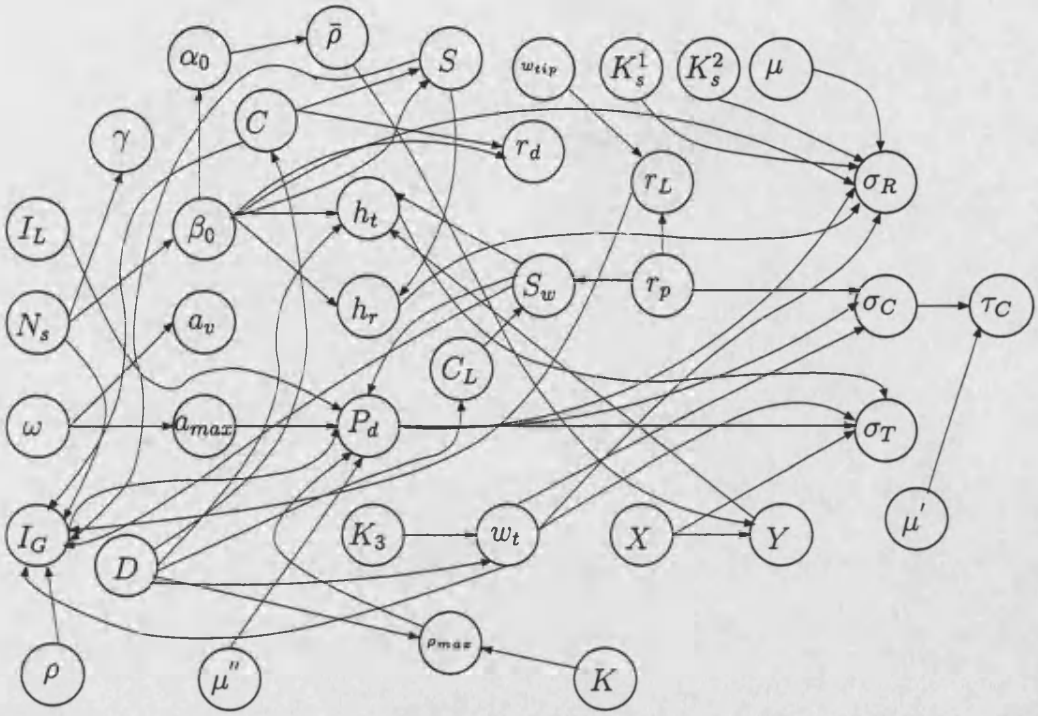


Figure 6.5: Design graph for the Geneva mechanism

in the Geneva mechanism design due to its critical influence on the life of the mechanism [Lee (1981)].

To make a choice for a controlling decision variable for the subsequent manufacturing process, the sensitivities of performance variables with respect to the decision variables should be considered. For a robust negotiation process the agents should choose the decision variable with least influence on performance variables. The following observations can be made from Table 6.4 with respect to the influences of the decision variables on τ_C.

- Increasing the wheel diameter (D) increases the shear stress (τ_C). The quantitative dependency is $\psi_{D,\tau_C} = 2.56811$. Thus an increase of wheel diameter by 10% would lead to a 25.6% increase in shear stress. An increase in D was not considered desirable. Additional reasons for not increasing the wheel diameter are (a) manufacturing limitations and (b) possibility of breakage at wheel root due to high sensitive influence of D on root stress,

Table 6.4: Effective qualitative and quantitative dependencies between design variables including the number of inter-nodal paths

Decision variables	Performance variables								
	τ_C			σ_R			σ_T		
D	$D \rightarrow \tau_C$			$D \rightarrow \sigma_R$			$D \rightarrow \sigma_T$		
	NP:	δ_{D,τ_C}	ψ_{D,τ_C}	NP:	δ_{D,σ_R}	ψ_{D,σ_R}	NP:	δ_{D,σ_T}	ψ_{D,σ_T}
	6	+	2.56811	7	+	24.4438	8	-	-2.14665
r_p	$r_p \rightarrow \tau_C$			$r_p \rightarrow \sigma_R$			$r_p \rightarrow \sigma_T$		
	NP:	δ_{r_p,τ_C}	ψ_{r_p,τ_C}	NP:	δ_{r_p,σ_R}	ψ_{r_p,σ_R}	NP:	δ_{r_p,σ_T}	ψ_{r_p,σ_T}
	3	-	-0.35064	2	-	-0.00120	3	-	-0.88780
S_w	$S_w \rightarrow \tau_C$			$S_w \rightarrow \sigma_R$			$S_w \rightarrow \sigma_T$		
	NP:	δ_{S_w,τ_C}	ψ_{S_w,τ_C}	NP:	δ_{S_w,σ_R}	ψ_{S_w,σ_R}	NP:	δ_{S_w,σ_T}	ψ_{S_w,σ_T}
	2	+	0.12838	2	-	-0.00135	3	-	-0.99350
C	$C \rightarrow \tau_C$			$C \rightarrow \sigma_R$			$C \rightarrow \sigma_T$		
	NP:	δ_{C,τ_C}	ψ_{C,τ_C}	NP:	δ_{C,σ_R}	ψ_{C,σ_R}	NP:	δ_{C,σ_T}	ψ_{C,σ_T}
	2	+	2.17024	3	+	24.4384	2	-	-3.0
w_{tip}	$w_{tip} \rightarrow \tau_C$			$w_{tip} \rightarrow \sigma_R$			$w_{tip} \rightarrow \sigma_T$		
	NP:	δ_{w_{tip},τ_C}	ψ_{w_{tip},τ_C}	NP:	$\delta_{w_{tip},\sigma_R}$	ψ_{w_{tip},σ_R}	NP:	$\delta_{w_{tip},\sigma_T}$	ψ_{w_{tip},σ_T}
	1	+	2.37219	1	-	-0.02498	1	-	-3.27936

σ_R.

- The pin radius (r_p) has a negative influence on the shear stress τ_C. r_p is the main decision variable of the driver agent and cannot be randomly increased without affecting the constraint on centre distance (C).

- The influence of centre distance (C) was found to have a positive influence on the state of shear stress (τ_C).

- An increase in the value of wheel tip width (w_{tip}) would also lead to a marked increase in the value of τ_C. Table 6.9 shows that increasing w_{tip} reduces root stress (σ_R) and tip stress (σ_T). However, w_{tip} could only be increased upto a certain limit without affecting the overall wheel size. Moreover, since w_{tip} is governed by the wheel agent alone, this choice is not in the common interest of both agents.

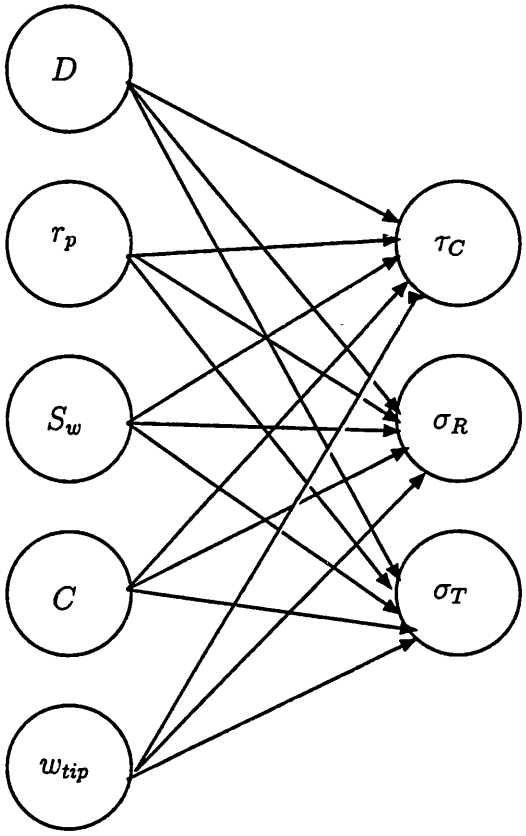


Figure 6.6: Reduced network between decision and performance variables

- Of the five decision variables, the influence of wheel slot width (S_w) on τ_C was appeared to be minimal. This is one common design parameter that can be explictly approached based on mathematical relations from both wheel and driver design perspectives.

Based on the above observation, the slot width (S_w) emerged as the negotiation variable and was considered a rational choice for the following key reasons:

- S_w can be approached from both driver and wheel perspectives
- Among the other influences considered ($C \rightarrow \tau_C$, $D \rightarrow \tau_C$, $w_{tip} \rightarrow \tau_C$, $r_p \rightarrow \tau_C$), the sensitivity of S_w on τ_C was found to be minimal.

6.7 Design for function

6.7.1 Initial design

Based on the design specification, the wheel- and driver agents evaluated their associated design variables as shown in Table 6.5. This also required the designer to make initial assumptions on the values of wheel diameter (D) and wheel tip (w_{tip}). For the purpose of generating utilities subsequently, alternative initial designs were generated for three different values of wheel diameter (D).

Table 6.5 Initial design results for the Geneva mechanism evaluated by the wheel and driver agents

Design variable type	Design variables	Units	Initial design 1	Initial design 2	Initial design 3
Specified variables	N_s	-	6	6	6
	ω	rpm	1000	1000	1000
	I_l	$Kg-m^2$	1.1298e-06	1.1298e-06	1.1298e-06
Assumed variables	D	m	0.05	0.06	0.07
	w_{tip}	m	0.005	0.005	0.005
Wheel agent: evaluated variables	C	m	0.0288	0.0346	0.0404
	C_L	m	1.27e-04	1.52e-04	1.77e-04
	I_g	$Kg-m^2$	2.1616e-06	3.6067e-06	5.0175e-06
	r_L	m	0.0058	0.008	0.0099
	a_{max}	rad/sec^2	14804.4	14804.4	14804.4
	σ_R	MPa	3.9	20.37	60.74
	σ_T	MPa	0.131	0.109	0.0891
	S_w^{wh}	m	0.0071	0.0085	0.0099
Driver agent: evaluated variables	S_w^{dr}	m	0.008	0.0094	0.0108
	P_d	N	3.1641	3.7944	4.2211
	r_p	m	0.0035	0.0042	0.0049
	τ_C	$Kg-m^2$	2.5	2.28	2.06
	r_d	m	0.0144	0.0173	0.0202

Note: S_w^{wh}, S_w^{dr} represent the slot width (S_w) calculated by the wheel and driver agents respectively

6.7.2 Identification of type 1 conflict

Table 6.5 shows that the evaluation of the slot width (S_w) from the perspective of the two software agents (wheel agent and driver agent) resulted in different values (0.0085 m and 0.0094 m respectively for initial design 2). Therefore Type 1 conflict occurred between the two software agents as shown in Figure 6.7, over the value of slot width[2]. The wheel diameter (D) and the pin radius (r_p) were chosen as the controlling decision variables for the wheel and driver agents.

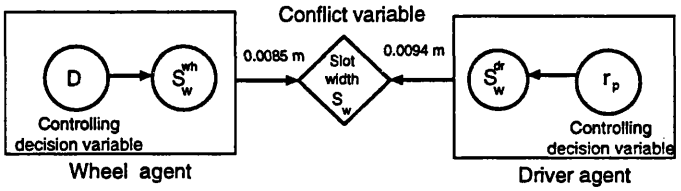


Figure 6.7: Conflict between wheel and driver agents on slot width, S_w

6.7.3 Formation of agent preferences in type 1 conflict

Based on the three different initial designs shown in Table 6.5, the utilities for wheel and driver agents were formed by varying the choices for controlling decision variables, D and r_p as shown in Figure 6.8.

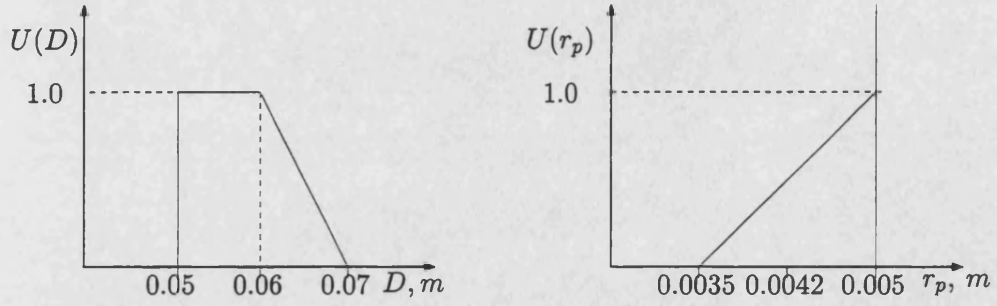
For the wheel agent, the utility function in Figure 6.8 (a) shows that the utility level ($U_1(D)$) remains constant in the range $(0.05-0.06)m$ and then shows a sharp decrease. With the objective of keeping σ_R to a minimum, the utility variation in the range $(0.05 - 0.06m)$ was considered reasonable. For the driver agent, the choice of utility was based on the key fact that a higher value of r_p would have a minimising influence the contact stress (τ_C). These utilities were specified to the design agent.

These utilities were propagated through the reduced design network to obtain the

[2]Based on initial design 2 in Table 6.5.

changes in the value of slot width (S_w) from each perspective and the resultant utilities were determined as shown in Figure 6.9 (a) and (b).

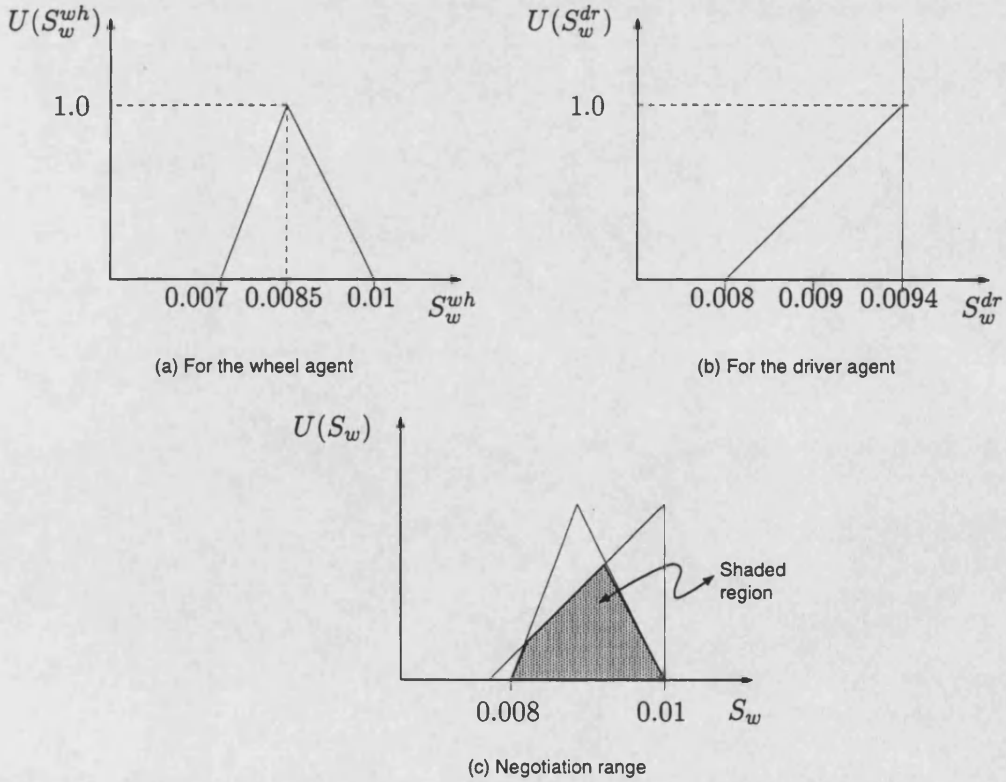
A negotiation range for slot width, common to both agents is identified under the shaded region in Figure 6.9 (c). The negotiation range represents the acceptable range of solution to both agents and its existence of negotiated solutions.



(a) For the wheel agent

(b) For the driver agent

Figure 6.8: Utility functions for the controlling decision variables



(a) For the wheel agent

(b) For the driver agent

(c) Negotiation range

Figure 6.9: Utility functions for the conflict variable

6.7.4 Negotiation of type 1 conflict

Based on the specified utilities in Figure 6.9, the negotiation agent evaluated four different game theoretic solutions [Figure 6.10]. A summary of the results of negotiation is presented in Table 6.6. The Kalai-Somordinsky solution is taken as the final negotiated settlement due its property of equal utilities to both the agents. For the present case the negotiated solution evaluated to a slot width of 0.009 m. The negotiated value of slot width S_w was propagated throughout the design network in order to obtain the revised design.

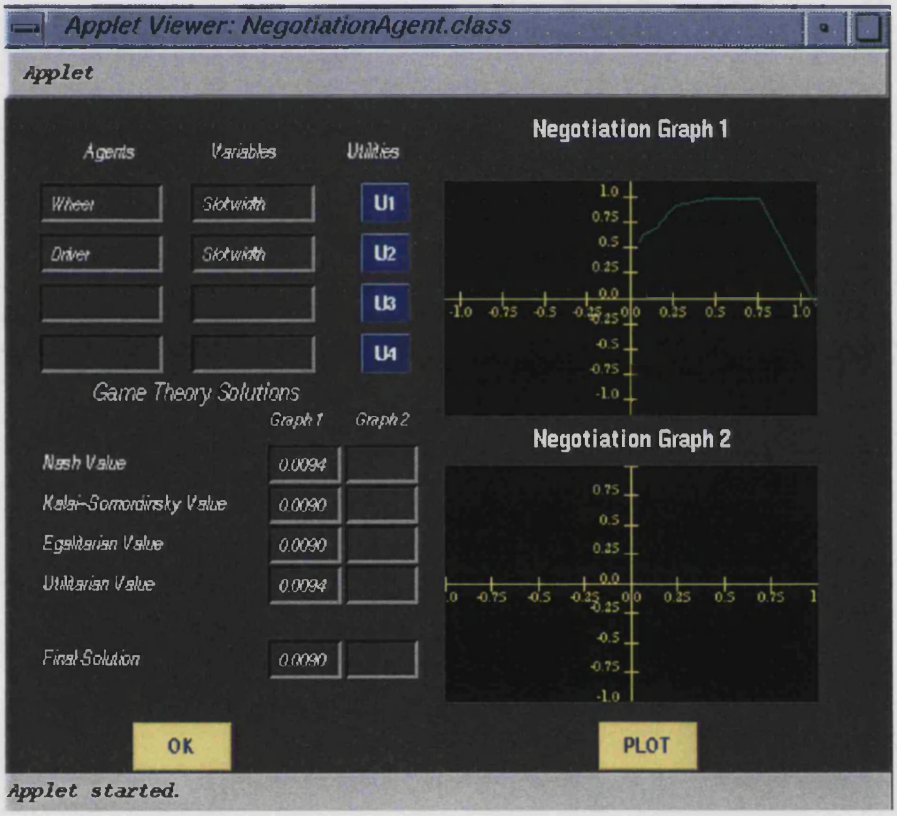


Figure 6.10: Utility curve for software-software agent conflict over the slot width

Figure 6.11 shows the neighbouring variables that affect S_w. Of these, C' and C'' represent clearance variables which can be varied from both wheel and driver design perspectives within a range (e.g., 0.00254 - 0.005 mm). The values for C' and C'' are changed by $+0.0005$ and -0.0004 respectively[3]. This completed the

[3]The idea of clearances presented in this study is based on the limits and fits for cylindrical

Table 6.6: Game theory solutions for conflict on slot width

Agents	Conflict variable	Preferred values	Game theory solutions				Final game solution	Negotiated value for S_w
			N(S)	K(S)	E(S)	U(S)		
Wheel agent	S_w, m	0.0085	0.0095	0.0090	0.0090	0.0095	0.0090	0.0090
			Utility values [no units]					
			0.6795	0.7566	0.7566	0.6795		
Driver agent		0.0094	0.0094	0.0090	0.0090	0.0094	0.0090	
			Utility values [no units]					
			1.0	0.7566	0.7566	1.0		

design for function stage and was followed by design for manufacture evaluation which is presented next.

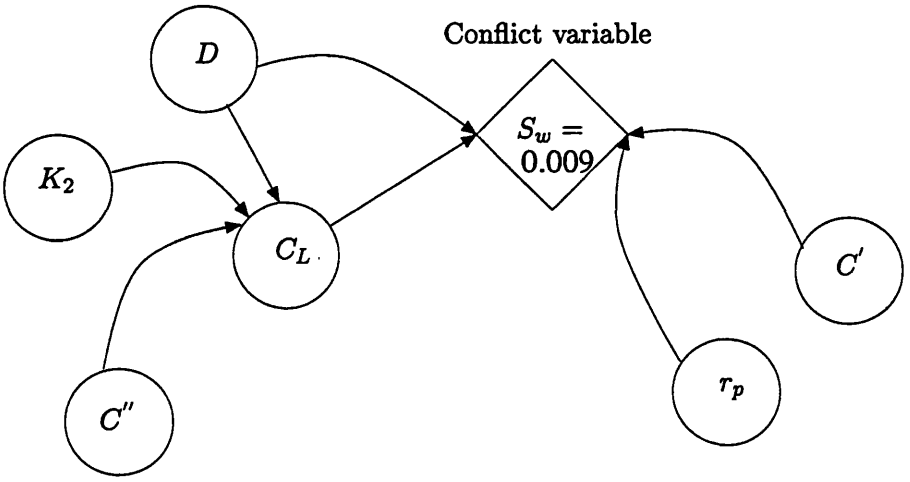


Figure 6.11: Design network pertaining to slot width

parts. In particular an approximate estimate is obtained from running and sliding fits [Hasty and Potts (1966), Shighley and Mitchell (1984), Lee (1981), Hill (1999)].

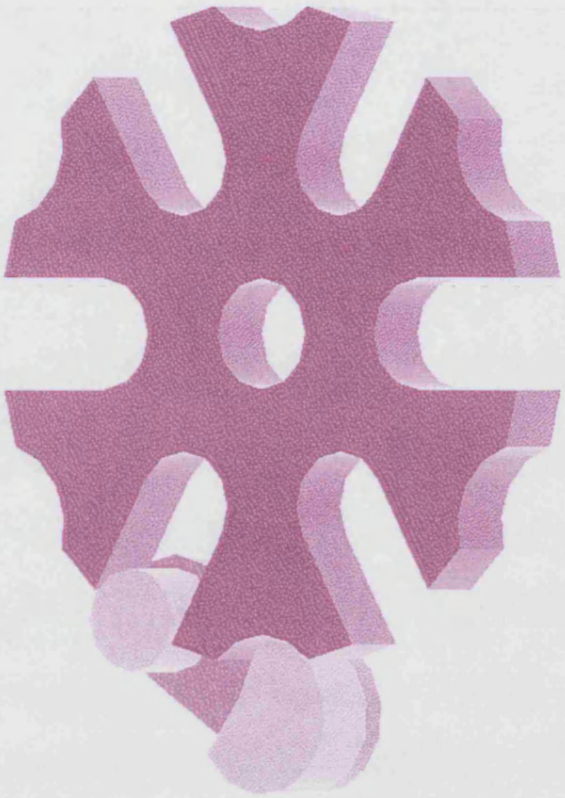


Figure 6.12: CAD model of Geneva wheel and driver

6.8 Design for Manufacture

6.8.1 Manufacturability analysis

The designer prepared the part drawings using AutoCAD based on the negotiated functional design. The drawings of the wheel and driver were generated as shown in the Figure 6.12. These drawings were then analysed for manufacturablity.

This analysis consisted of the following key stages:

- Examine the part drawings;
- Analyse the design's mouldability;

- Propose any modifications in design, if necessary;
- Form a preliminary mould design.

6.8.2 Identification of type 2 conflict

The machinists (HA3 and HA4) examined the part drawings based on its dimensions and limits set by the design specification. Although all geometrical and material constraints were identified prior to the initial design, there were some resource restrictions on machine tools such as the availability and sizes of milling cutters which could affect the depth of cut and therefore the wheel thickness (w_t). Thus w_t emerged as a variable of common interest between the wheel agent and the machinists.

Type 2 conflicts occur when a software and human agent differ in their choice for the value of the common variable, which in this case was wheel thickness w_t. The machinists suggested a reduction in wheel thickness by approximately 25% of the value evaluated by the wheel agent. This need for thickness reduction was due to a possibility of incomplete edge formation in the manufactured product. As shown in Figure 6.13, a drill tool with diameter greater than $2mm$ would result in rounded edges thus affecting the tip width and pin entry. A drill tool with less than $2mm$ diameter was also not preferred by the machinist agents due to the high depth of cut required. As a result a conflict was detected between the wheel design agent and the human agents on the value of wheel thickness (w_t) [Figure 6.14].

The wheel diameter (D) was retained (from type 1 conflict) as the controlling decision variable for wheel agent. The following present the key reasons for this choice:

- Three different values for wheel thickness (w_t) were obtained by varying the

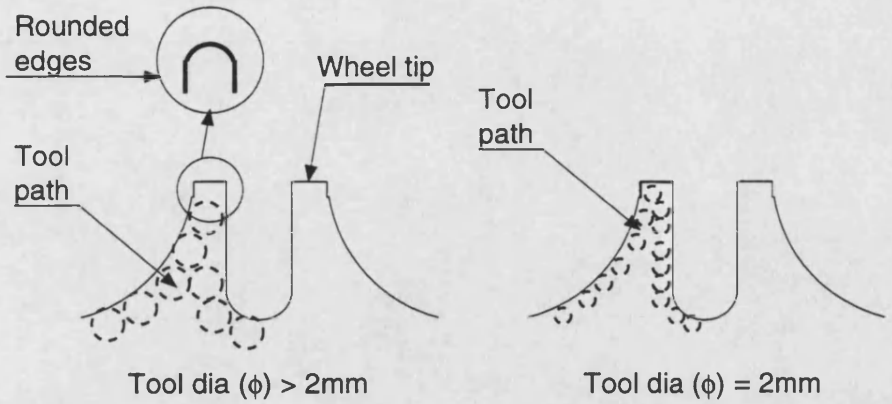


Figure 6.13: Mould manufacturability: effect of drill tool on wheel tip width

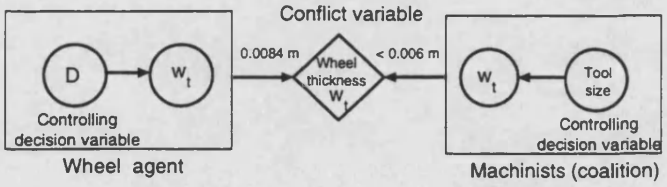


Figure 6.14: Conflict between wheel and machinists on wheel thickness, w_t

wheel diameter (D) during the initial design.

- Wheel diameter (D) directly influences[4] the wheel thickness (w_t) as shown in the Table 6.3.

The machinists chose the size of milling tools as the controlling decision variable. This choice was due to the following key reasons:

- Availability of milling tools
- Depth of cut required
- Safety of the manufacturing operation
- Prevention of rounded wheel tip

[4]The other variable in addition to D which influence the wheel thickness is K_3. K_3 is not considered as the controlling decision variable due to the restrictions in its variation [Johnson, (1956)].

6.8.3 Formation of agent preferences in type 2 conflict

The wheel thickness was the conflict variable in this case. For the wheel agent, the controlling decision variable was the wheel diameter (D) and the utility $U(D)$, same as in type 1 conflict, Figure 6.8(a). The wheel agent's utility $U_1(w_t)$ for conflict variable (w_t) was then derived from $U_1(D)$, as shown in Figure 6.15(a). As can be observed from the figure, the utility shows an increasing trend with any increase in the value of wheel thickness (w_t). This trend is consistent with the influence of w_t on the performance parameter τ_C as shown in Table 6.7.

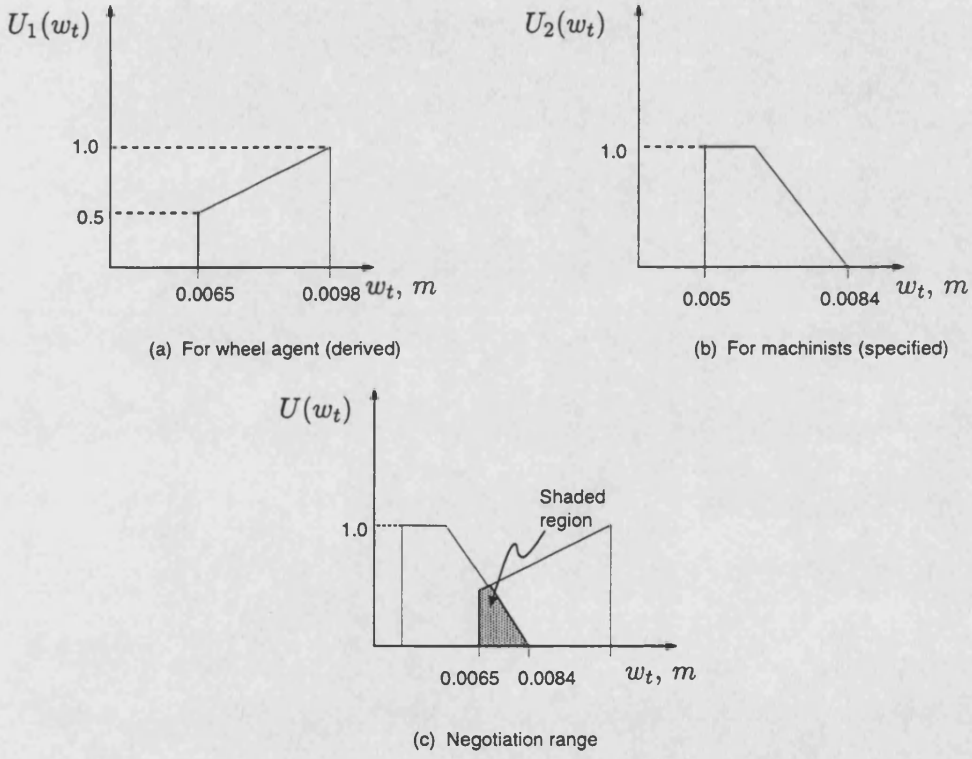


Figure 6.15: Utility functions for the conflict variable wheel thickness (w_t) in type 2 conflict

The machinists specified their utility description directly for the conflict variable w_t as shown in [Figure 6.15(b)]. The Figure shows that the best preferred range for the machinists is between 0.005 - 0.006 m. Therefore any value greater than 0.006 m shows a decrease in the utility for the machinists. The key reason for such a variation in the utility was attributed to the availability of the size of milling tools (typically 2 mm) which would ensure a safe milling operation. However, it

Table 6.7: Effect of wheel thickness on stress levels

Effective variable dependencies								
$w_t \rightarrow \sigma_R$			$w_t \rightarrow \sigma_T$			$w_t \rightarrow \tau_C$		
NP:	δ_{w_t,σ_R}	ψ_{w_t,σ_R}	NP:	δ_{w_t,σ_T}	ψ_{w_t,σ_T}	NP:	δ_{w_t,τ_C}	ψ_{w_t,τ_C}
2	+	0.0036	2	-	-2.4275	2	+	0.5625

should be noted that a more accurate representation of utility in Figure 6.15(b) is possible and could be generated by performing a tool-breakage analysis. Such an analysis would include several variables such as tool material, cutting speed, depth of cut and work piece material. for this study, such an elaborate analysis was not considered necessary to determine the theoretical utility functions for the machinists. Figure 6.15(c) shows the negotiation range which ensures a negotiated settlement. The designer specified these utilities of the wheel agent and the machinists to the negotiation agent.

6.8.4 Negotiation of type 2 conflict

The utilities in Figure 6.15 were input to the negotiation agent which constructed the joint utility curves for the two agents as shown in Figure 6.16. The four negotiated solutions are shown in Table 6.8. Table indicates that the Kalai-Somordinsky value which yielded a utility of 0.59 to both agents was chosen as the final settlement. The negotiated value for the wheel thickness was found out to be 0.007 m.

The negotiated value for wheel thickness was propagated through the design network to obtain an updated design. As shown in Figure 6.17, a sub-network of all variables relating to wheel thickness was formed to this end.

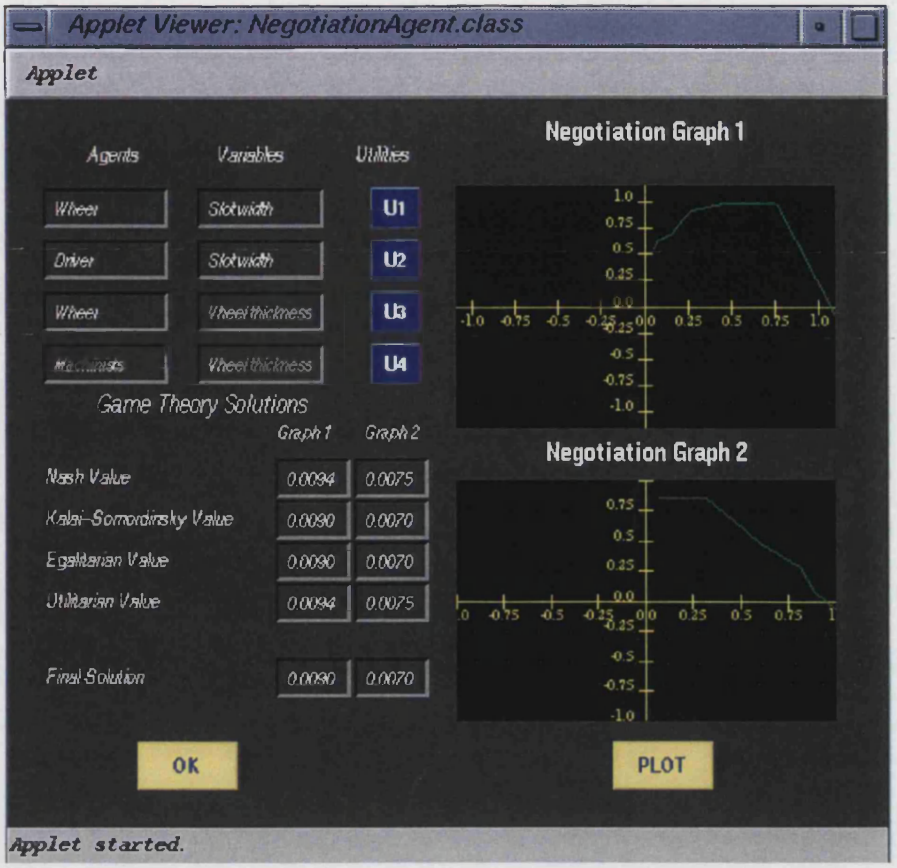


Figure 6.16: Utility curves for software-human agent conflict over wheel thickness

The change in w_t was propagated to K_3 to enable the restoration of D to the original functional design as far as possible. Since K_3[5] is is directly influences w_t, this was an apt choice. The corrected value of K_3 was found to be 0.1167 as indicated in Figure 6.17. As a result of the changes occurred in the values of S_w and w_t, an updated design was evaluated. Only the performance parameters (such as stress and load) were altered and not the key geometric parameters such as the wheel diameter and wheel tip width. The finalised design served as a basis for the design of the mould which is discussed next.

[5]The suggested range of K_3 is between 0.05 to 0.14 [Johnson (1956)].

Table 6.8: Game theory solutions for conflict on wheel thickness

Agents	Conflict variable	Preferred values	Game theory solutions				Final game solution	Negotiated value for w_t, m
			N(S)	K(S)	E(S)	U(S)		
Wheel agent	w_t	0.0084	0.0075	0.0070	0.0070	0.0075	0.0070	0.0070
			Utility values [no units]					
			0.6884	0 .5899	0.5899	0.6884		
Machinists		0.0060	0.0080	0.0070	0.0070	0.0080	0.0070	
			Utility values [no units]					
			0.5161	0.5899	0.5899	0.5161		

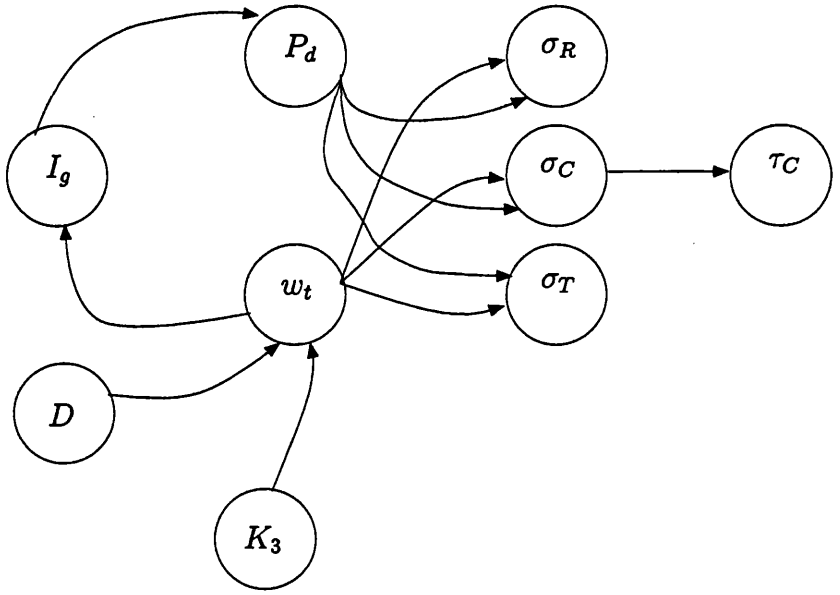


Figure 6.17: Design sub-network pertaining to wheel thickness

6.9 Development of the manufacturing tool

6.9.1 Mould design

Based on the initial design, the designer HA1 created the drawings of the mould using AutoCAD. The mould drawings for wheel and driver without the positioning of the gates were generated with a $0.5mm$ allowance for shrinkage [6] as shown in Figure 6.18. Three different designs were suggested by the designer based on

[6]This allowance for polypropylene was assumed based on the information provided by GE Plastics, Inc. USA. http://www.geplastics.com

the following key considerations:

- Material flow characteristics;
- Shape of the product;
- Ease of manufacture, time and cost;
- Ease of ejection.

Initial design of the mould was carried out considering the flow characteristics (e.g., uniform pressure, symmetrical flow) and ease of operation (e.g., ejection). Flow characteristics would depend on the type of plastic material used, path of mould flow and injection parameters such as pressure, temperature and volume. Therefore machinists' (HA3 and HA4) main consideration was to design a mould that would offer least resistance to the fluid flow. The machinists came up with a two-plate mould as shown in Figure 6.19.

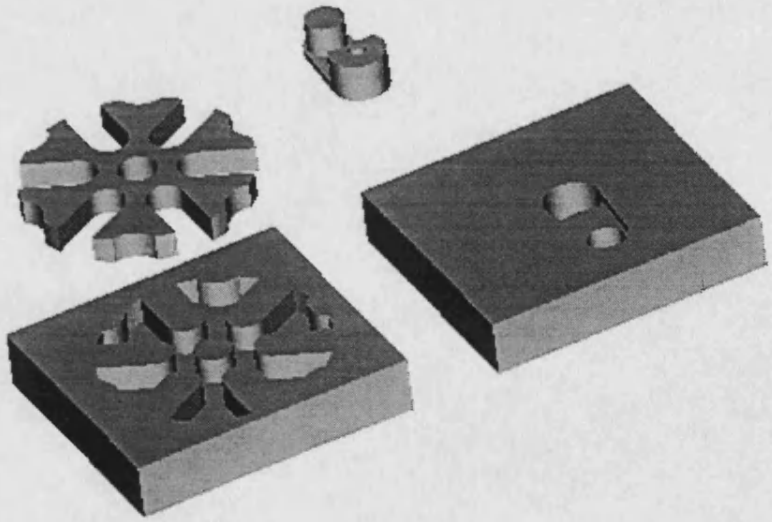


Figure 6.18: AutoCAD model of product and the mould

The mould designer (HA2) examined the mould design put forward by the machinists and pointed out two distinct drawbacks:

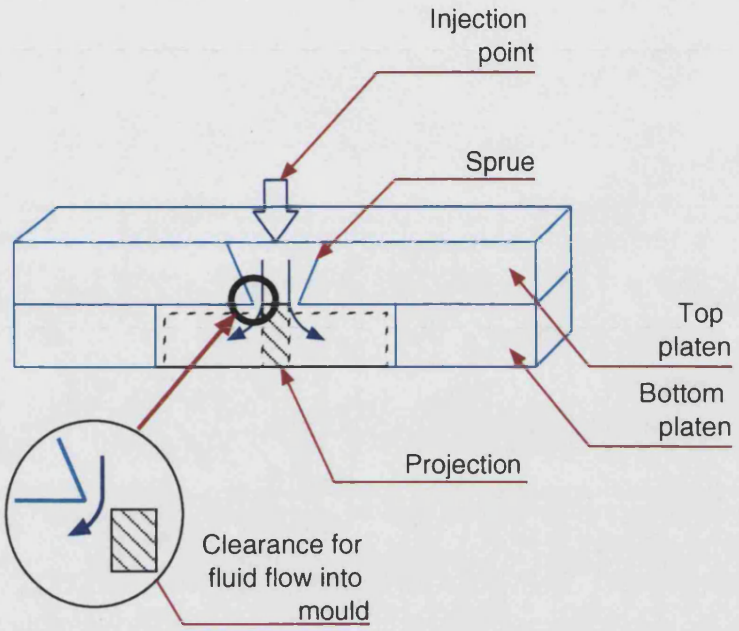


Figure 6.19: Initial design of mould

- Need for very high injection pressure since the clearance for fluid flow was too small (of the order of 0.5 mm);
- Covering of plastic over the projection which would have made ejection operation difficult.

There were additional concerns such as formation of blow-holes due to a very high injection pressure. Therefore it was agreed between the mould designer and the machinists' to modify the mould design as shown in Figure 6.20. This however, presented drawbacks such as machining the slot at the centre without affecting the critical tolerances (e.g., central hole clearance between wheel and mounting base). Therefore, the mould designer proposed a refined design which is shown in Figure 6.21. The key refinements in the design are as follows:

- ease of plastic flow;
- wheel dimensions remain unaffected;
- symmetry in injection.

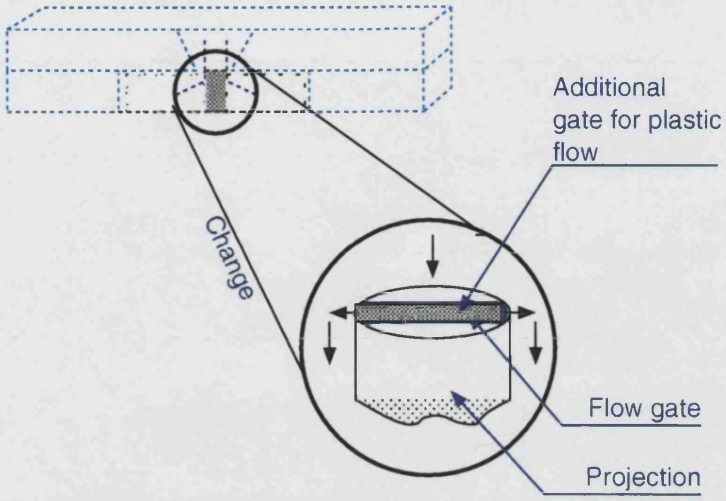


Figure 6.20: Suggested change in the initial design

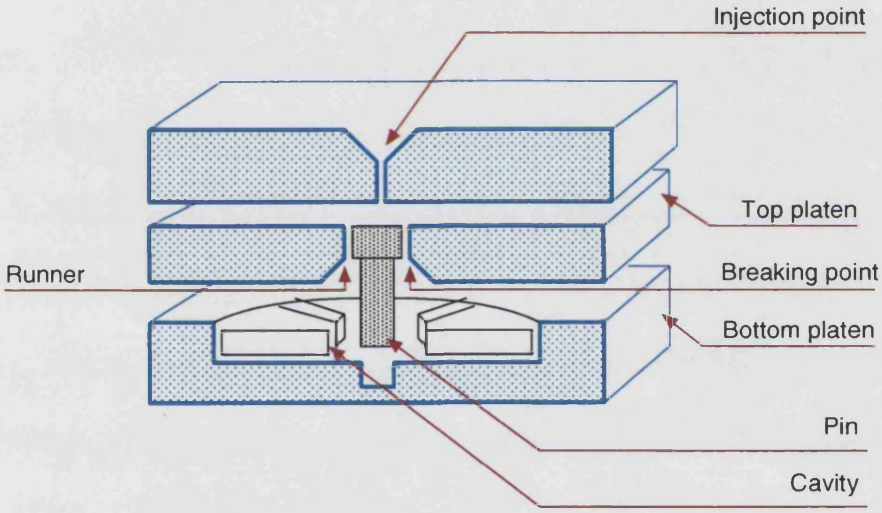


Figure 6.21: Final design of the wheel mould

6.9.2 Identification of type 3 conflict

The above design satisfied the general requirements of a feasible mould stated above and was thus finalised. However, there was still need for an ejection mechanism[7]. Based on the brainstorming discussions between the machinists (HA3 and HA4) and designers (HA1 and HA2), it was decided that ejector pins would need to be arranged in the base of the mould. The designers suggested an ejec-

[7]Several test-runs of product manufacturing were carried out to examine the ease of ejection without a separate ejector mechanism

tion mechanism consisting of six ejector pins located symmetrically as shown in Figure 6.22(a). The machinists, however, suggested the use of two pins in opposite corners of the mould (also in symmetrical locations) as shown in Figure 6.22(b). Thus a type 3 conflict was detected between human agents on the required number of pins as shown in Figure 6.23.

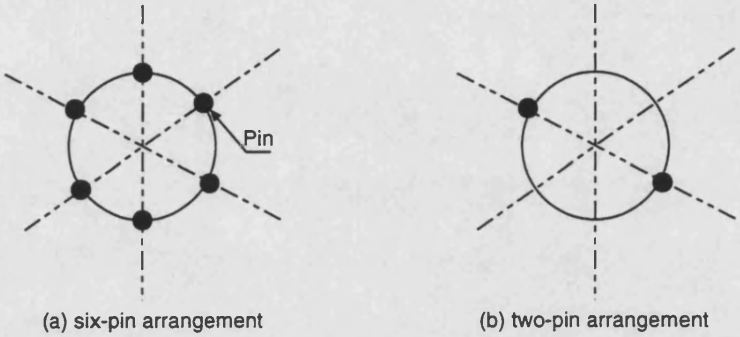


Figure 6.22: Pin configuration in the ejection mechanism

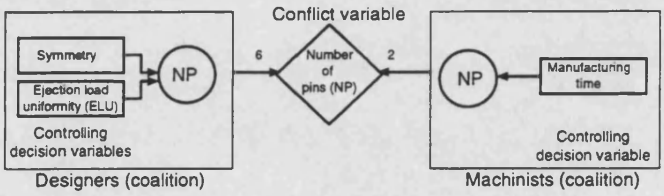


Figure 6.23: Conflict between designers and machinists on the number of ejector pins, NP

6.9.3 Formation of agent preferences in type 3 conflict

The designers identified their preferences on the required number of pins based on the necessity of the symmetry of pin-arrangement and uniformity of ejection load. Symmetric arrangement of pins would lead to symmetric loads and hence reduce warping or twisting of the product during ejection. Therefore the symmetry and uniformity of load were chosen as the controlling decision variables to define the designers' preferences.

Figure 6.24 (a) shows the variation of utility $U_1(symmetry)$ of pin-symmetry for the designers. This variation shows that any symmetric arrangement of pins

would correspond to a utility of 1.0 whereas for asymmetric cases (0 and 5 pins) the utility is zero.

Figure 6.24 (b) shows the utility $U_1(ELU)$ of the ejection load uniformity for the designers. The maximum utility of 1.0 would be attained when the number of pins is very large. The two utilities of the controlling decision variables were used as shown in Table 6.9 to form the utility of the number of pins from the designers' perspective as shown in Figure 6.24 (c). The necessity of pin-symmetry *and* ejection load uniformity means that the net utility $U_1(NP)$ is a minimum of the two component utilities.

Table 6.9: Utility of number of pins for the designers

Number of pins (NP)	Symmetry	U_1 (Symmetry)	Ejection load uniformity (ELU)	U_1 (ELU)	U_1 (NP) = Minimum [U_1 (Symmetry), U_1 (ELU)]
0	0	0	0	0	0
1	0	0	0.15	0.15	0
2	1	1	0.30	0.30	0.30
3	1	1	0.45	0.45	0.45
4	1	1	0.60	0.60	0.60
5	0	0	0.75	0.75	0
6	1	0	0.90	0.90	0.90

As for the machinists, the controlling decision variable was the manufacturing time which is the sum of the set-up time and machining time. For a given work-piece, the set-up time would remain constant whereas machining time will increase proportionally with the number of pins, as shown in Figure 6.25(b). The manufacturing time utility is shown in Figure 6.25(a) which indicates a uniform decrease with an increase in the manufacturing time. The machinists' utility for the number of pins is therefore obtained by calculating the total manufacturing time for different number of pins as shown in Table 6.10 and Figure 6.25(c).

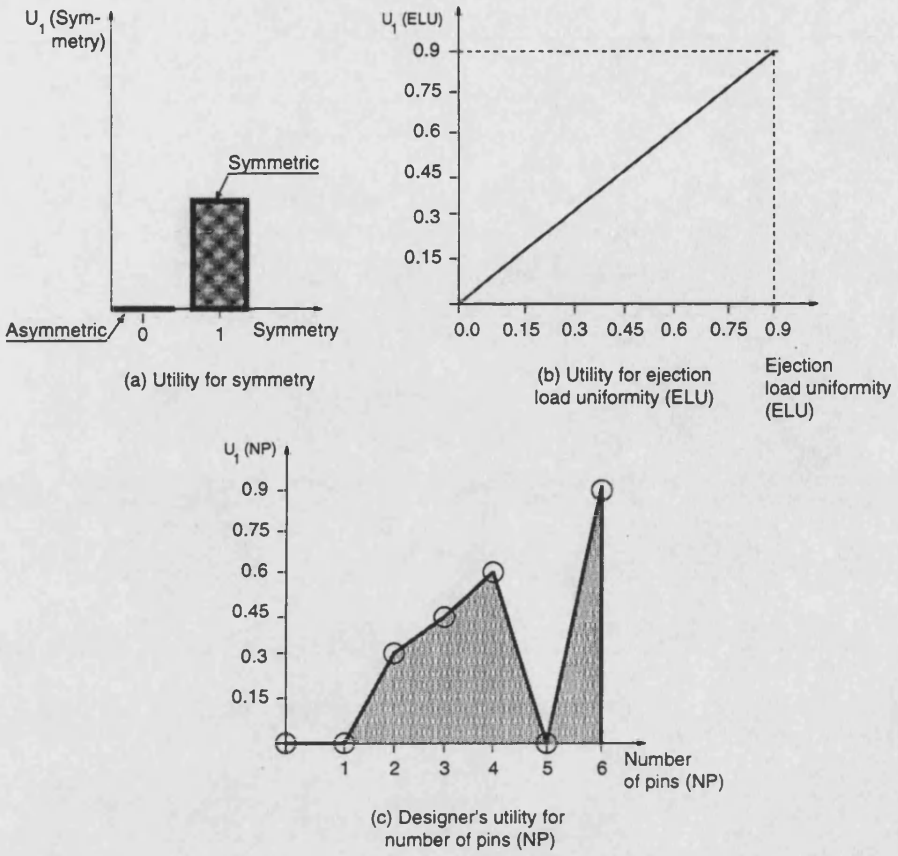


Figure 6.24: Designers' utilities for controlling decision variables and conflict variable

6.9.4 Negotiation of type 3 conflict

The utilities of the number of pins for the designers and the machinists [Figure 6.24 (c) and Figure 6.25 (c)] were input to the negotiation agent which constructed the joint utility curve for the two agents as shown in Figure 6.26. Table 6.11 shows the four game theory solutions obtained for type 3 conflict. It indicates that the Kalai-Somordinsky solution yielded equal utility of 0.341 to both agents. The other solutions, Nash and Utilitarian, however, evaluated a negotiated solution which gave the designers their maximum attainable utility of 0.9. Consistent with previous negotiation, the Kalai-Somordinsky solution was accepted as the final settlement of the conflict. It should be noted that the number of pins 2.4 corresponding to the Kalai-Somordinsky solution of 0.341 was rounded off to the nearest integer 2, to obtain the negotiated value for the number of pins.

Table 6.10: Utility of number of pins for the machinists

Number of pins (NP)	Set-up time (min)	Machining time (min)	Total manufacturing time (TM) (min)	U_2 (TM)	U_2 (NP)
0	0	0	0	1.0	1.0
1	20	3	23	0.41	0.41
2	20	6	26	0.333	0.333
3	20	9	29	0.256	0.256
4	20	12	32	0.174	0.174
5	20	15	35	0.102	0.102
6	20	18	38	0.025	0.025

6.9.5 Mould manufacturing

The mould for the wheel and driver of the Geneva mechanism was manufactured in aluminium. The machinists (HA3 and HA4) prepared the aluminium blocks for manufacturing the mould by using the Matchmaker CNC milling machine. A flat mill tool was used to cut the blocks to the required dimensions as shown in Figure 6.27.

Once the aluminium blocks were ready for mould making, HA3 prepared the manufacturing drawings for the two parts (wheel and driver) using SmartCAM based on the design drawings and .DXF files supplied by HA1. SmartCAM did not provide a perfect interface for .DXF files (generated by AutoCAD 2000) and lost some information in the design files. Therefore, HA3 had to regenerate some of the lost profiles. The SmartCAM generated an NC-code (shown in Appendix B) which was directly input to the milling machine. The machining process was initiated and the manufacturing process was largely automated. However, the machinists had to manually adjust the coolant position throughout the operation. There were also instances such as controlling the cutting rate depth, to prevent tool chatter and breakage problems, where HA3 reacted in a pro-active manner. Once the machining operation was complete, the blocks were removed. On inspection, the machined mould [shown in Figure 6.28 and Figure 6.29] was

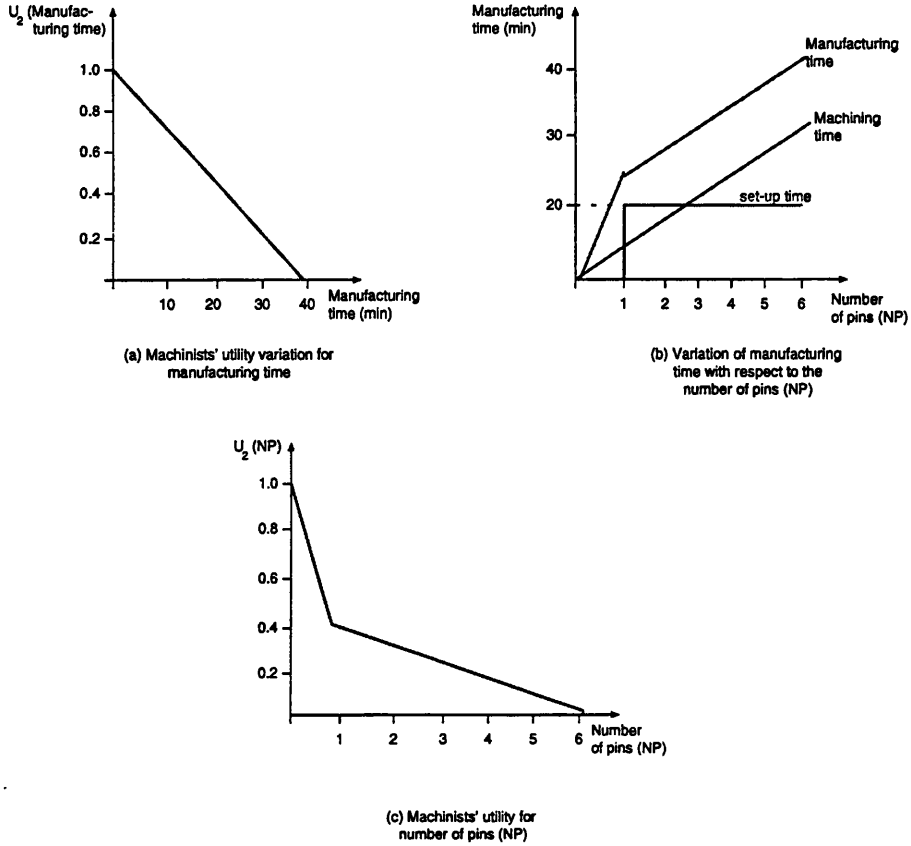


Figure 6.25: Machinists' utilities for controlling decision variables and conflict variable

found to be dimensionally accurate.

6.10 Product manufacturing

6.10.1 Initial manufacturing

The key stages involved during the initial manufacture of the product were:

- Mounting of moulds onto the injection moulding machine platens;
- Setting the machine parameters (injection pressure, temperature and cooling time);

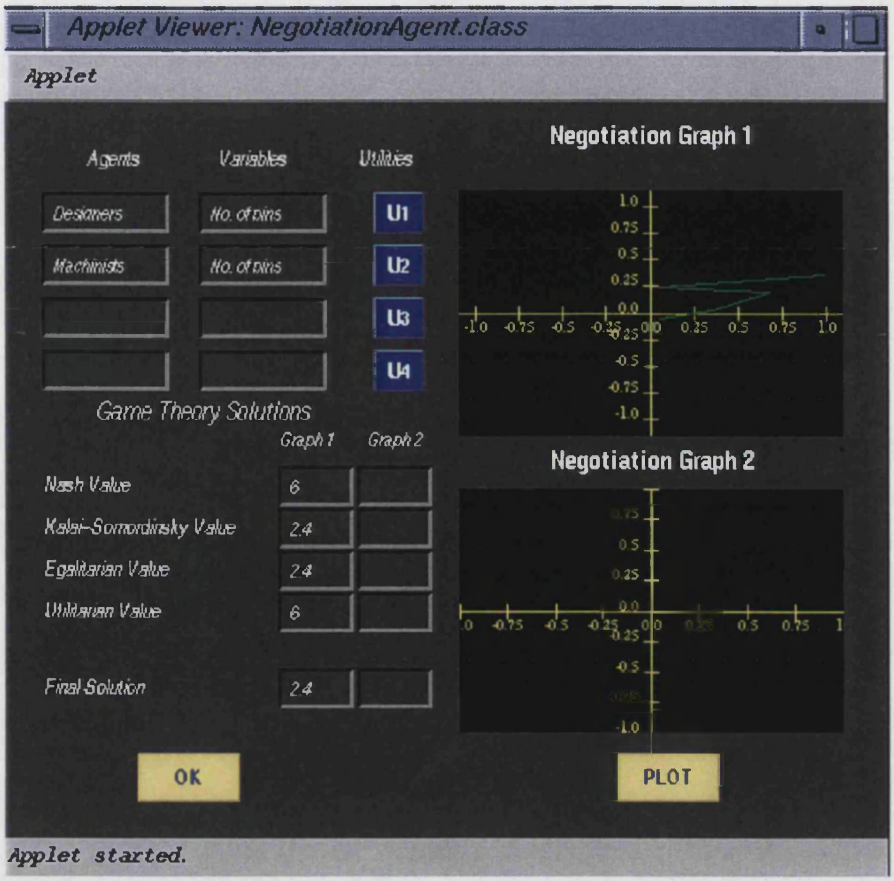


Figure 6.26: Utility curves for human-human agent conflict over number of ejector pins

- Moulding operation;
- Ejection of product from the mould.

6.10.2 Reccurence of type 3 conflict and its resolution

The two-pin ejection mechanism (presented in section 6.4.7) proved to be ineffective and often led to warping and twisting of the moulded wheel. The decision made by the human agents initially choose two-pins for the ejection mechanism was found unsatisfactory. The alternative option to use all six pins (as evaluated by the Nash and Utilitarian solutions in Table 6.11) was then accepted. The additional pins were manufactured and were then fitted into the moulds as shown

Table 6.11: Game theory solutions for type 3 conflict on the number of ejector pins

Agent coalition	Conflict variable	Preferred values	Game theory solutions				Final game solution	Negotiated value for the number of pins (NP)
			N(S)	K(S)	E(S)	U(S)		
Human designers (HA1 and HA2)	Number of pins (NP)	2	6	2.4	2.4	6	2.4	$2.4 \cong 2$
			Utility values					
			0.9	0.34	0.34	0.9		
Human designers (HA3 and HA4)		6	2	2.4	2.4	2		
			Utility values					
			0.4	0.34	0.34	0.4		

Figure 6.27: Instance of the milling machine operation showing the workpiece

in the Figure 6.30.

The moulding process was carried out without any changes in the machine settings. The decision to use a six-pin based ejection mechanism for the wheel was found to be effective as the pins safely ejected the product. For the driver mould, however, the earlier two-pin arrangement was found sufficient and no changes were needed.

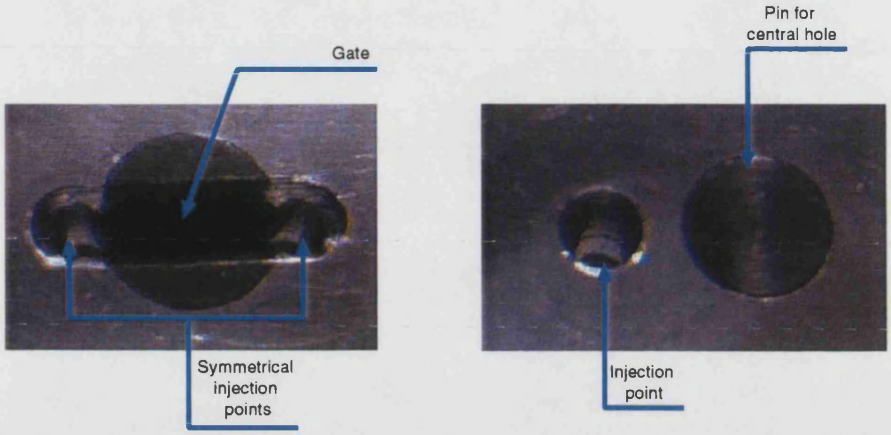


Figure 6.28: The injection points in Geneva wheel and driver moulds

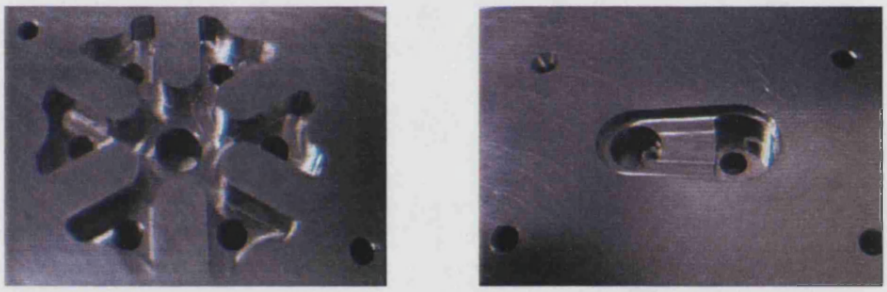


Figure 6.29: The Geneva wheel and driver moulds

6.10.3 Final product manufacture

The new mould for the Geneva wheel was used to manufacture the mechanism and was checked for its dimensional accuracy. The cooling patterns indicated a higher shrinkage in the thickness direction than expected during the design ($w_t \approx 6.4mm$).

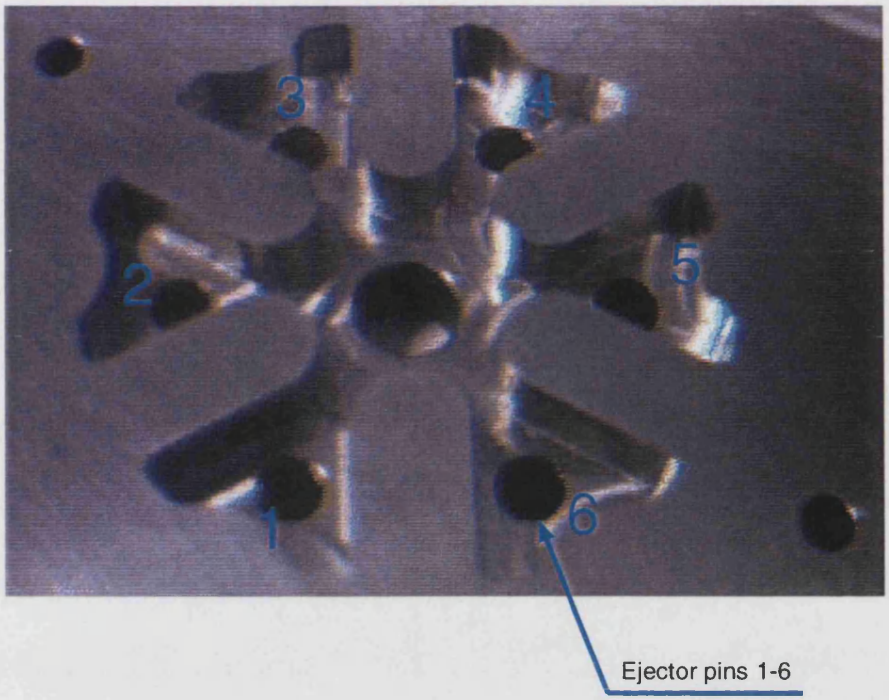


Figure 6.30: Ejector pins in the Geneva wheel mould

6.11 Experimental analysis

6.11.1 Product analysis

The manufactured product was assembled on to an aluminium base with a specified centre distance [Figure 6.31]. The assembled mechanism was functionally tested for its slot entry, exit and other prescribed motions [Figure 6.32]. The performance of the mechanism itself was found to be satisfactory for some arbitrary motions. The product exhibited fair shrinkage characteristic throughout and the clearances such as slot width, etc, were found to be well preserved.

6.11.2 Process analysis

Communication and archiving

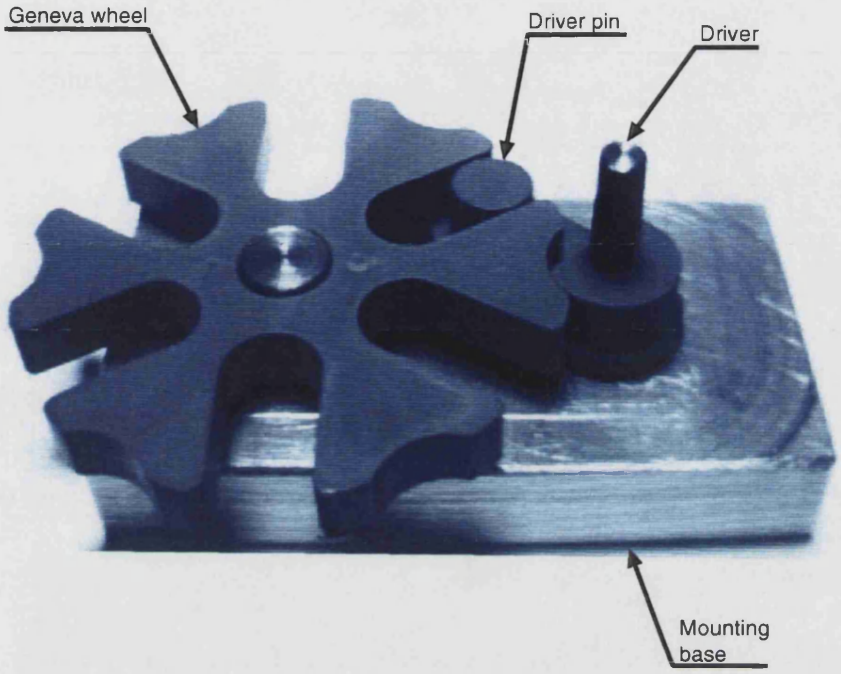


Figure 6.31: The assembled Geneva mechanism

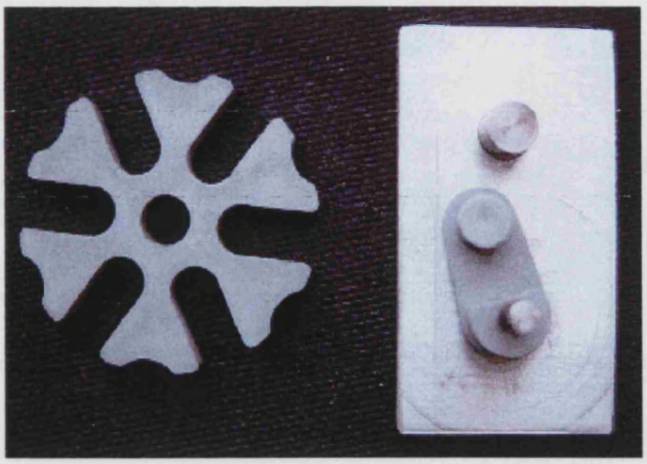
There were synchronous and asynchronous communication between agents (both software and human). As for archival, WWWeasel[8] and video capture tools were predominantly used though session proceedings were also archived off-line as paper-based documents.

Information sharing

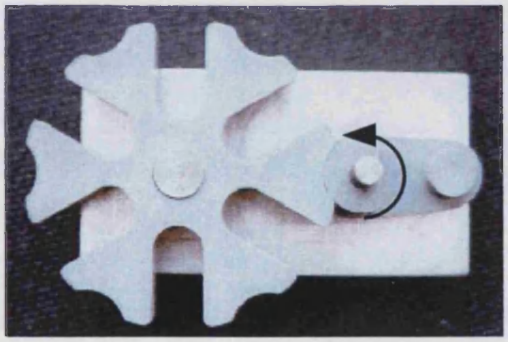
There was a variety of ways in which information was shared between agents:

- use of AutoCAD 2000 to generate and share web compatible CAD models;
- conversion of AutoCAD 2000 drawings to VRML and Inventor formats using Solid works;
- generation and sharing of the CAD drawings theoretically compatible with SmartCAM;

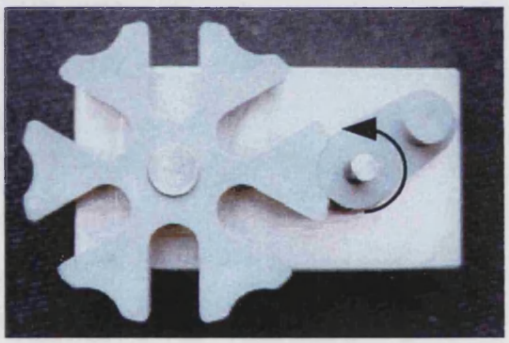
[8]Enterprise Integration Technologies, Inc. USA.



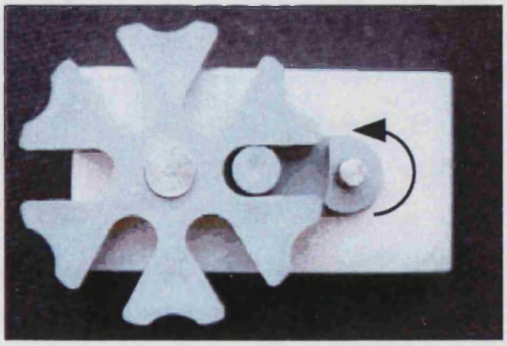
Moulded Geneva wheel, driver and aluminium base



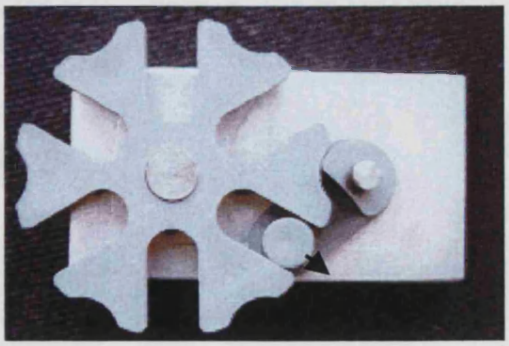
Wheel - fully locked position



Driver at an arbitrary position



Driver - fully engaed position



Driver - exit position

Figure 6.32: Prescribed motions for the assembled Geneva mechanism

- visualisation and manipulation of 3D CAD models in the shared whiteboard.

The sharing and joint manipulation of 3D CAD models in the whiteboard was useful when the designed product had to be evaluated with respect to its mouldability. For example, the 3D CAD models were imported into the whiteboard and the designers (HA1 and HA2) can manipulate a model (rotate, translate, etc.) such as the Geneva wheel as shown in Figure 6.33.

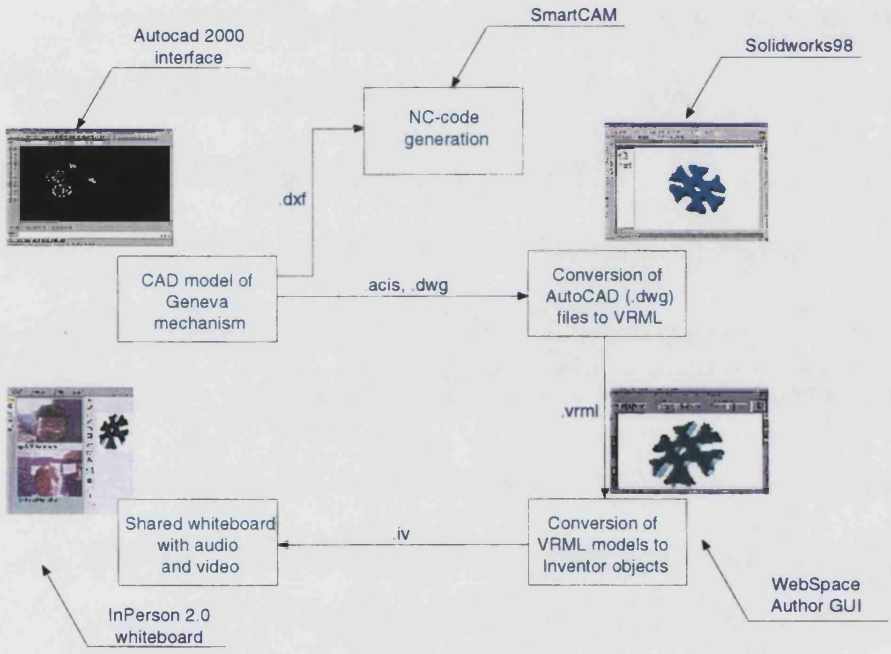


Figure 6.33: Information sharing between agents

6.12 Discussion

The collaborative design and manufacturing case study of the Geneva mechanism examined the following key aspects:

- The need for a mechanism to ensure the alignment of goals in a decision process where agents have preferences. Such a conflict resolution mechanism

needs to be flexible enough to deal with conflicts between heterogenous agents (both human and software);

- The agent synthesis and goal alignment hypotheses in a broader context of the design and manufacturing process.

6.12.1 Conflict resolution between agents

This case study viewed the conflict resolution between agents in product development as a key issue. It investigated the conflicts due differing preferences that occurred between software agents and humans through the development of a novel scheme for conflict resolution. The scheme (developed as a combination of game theory and dependency based reasoning) has been found to offer consistent design solutions while preserving the preferences of agents represented as utility functions. The conflict resolution between agents has been approached as a two-player game with specified utility functions. With respect to the chosen domain (in the class of parametric design) where parametric dependencies exist, the present results have justified the combination of dependency and game theory approach. However, this claim excludes possible situations where only fuzzy relations are established. The author's approach is compared with other techniques adopted for conflict resolution. The criteria for such a comparison were identified in Chapter 4.

Based on the quantitative results presented in Section 6.4.4, the Geneva mechanism presented a problem of higher complexity which has involved several more variables as compared to cases presented in Kanappan and Marshak (1993), Kusiak and Wang (1995) and Berker and Brown (1996). The author's approach has been validated in the context of parametric design problems with well defined analytical relations. The game theory based approaches need the specification of objective or utility functions. Therefore, the genericity of the author's approach can be seen at par with that of Kusiak and Wang (1995) and Kanappan

and Marshek (1993). Klien (1993) presented more generic schemes (mainly rule-based) for conflict resolution that are independent of the domain and hence the genericity of Klien's approach is high [9].

The importance of consistency of design solutions has been demonstrated via qualitative and quantitative reasoning. The reasoning based approach ensured that the solutions are consistent and updated for any change. The design of Geneva mechanism showed this in instances such as the effect of the slot width (S_w) or stress levels (τ_C). With respect to design consistency, the author's approach is superior to that of Kanappan and Marshek (1993) or Berker and Brown (1996). This is due to the fact that they did not consider the consistency of solutions in their negotiation schemes.

Some of the previous research approaches have focused on applying standard techniques based on game theory to evaluate a set of negotiated solutions [for example, see Kanappan and Marshek (1993)]. On the other hand, a weak approach of iterative contraction of solution space is adopted by Kusiak and Wang (1995) for obtaining a single point solution for a specific case. Hence most of the previous research has not focused on obtaining a single point solution nor alternative solution based on rational criteria. The author's research offers additional rules for isolating a single solution from a set of game-theoretic outcomes.

The communication between agents has been effected via formal and informal means. The aspect of formal KQML-based communication between software agents (in the context of negotiation) has been investigated in-depth by Berker and Brown (1996) and Frost and Cutkosky (1996). The research of Kanappan and Marshek and of Kusiak and Wang did not present any mechanisms for communication. The author's research has acknowledged the importance of communication in negotiation but this is not the main issue in the author's research.

[9] However, Klien (1993) did not quantify the results obtained from his generic schemes applied to a case study. This is not surprising since the schemes developed in his study were independent of the domain which bears a large proportion of the quantitative information.

However, the off-the-shelf communication tools have been used extensively.

With respect to a scheme for conflict resolution, the author's approach has developed, implemented and validated a novel scheme. Several of the previous research including that of Berker and Brown (1996) and Kanappan and Marshek (1993) did not systematically analyse their negotiation protocol. For example, Kanappan and Marshek obtained negotiated solutions using game-theoretic schemes but did not consider the effect of parametric dependencies nor did they analyse the quality of negotiated solutions. Kusiak and Wang (1995, 1996), however, presented a formal dependency based procedure for reasoning in a design network with an exception that their procedure is not as elaborate as compared to the author's research.

As seen in real-world scenarios, the author's research takes a view that conflicts always occur due to differing preferences and therefore total cooperation between agents cannot be guaranteed. Such a view was not adopted in Berker and Brown (1996) and Kusiak and Wang (1996) where the existence of total cooperation was assumed. Finally, the author's research and other approaches including that of Kanappan and Marshek (1993) have been applied to conflict situations involving only two agents or two coalitions. The conflicts between n-agents or n-coalitions, though an interesting research issue in game theory, has not been investigated by the author in the identified scope of this research.

The following key points emerge from this discussion:

- In a collaborative decision process, there is need for representation of agents' preferences;
- For conflict resolution between agents, it is necessary to combine the game theory and dependency based reasoning techniques. Such a combination has been shown to yield a rational single-point consistent solution.

6.12.2 Agent characteristics realised

The collaborative process in this case study consists of four software agents: Geneva mechanism design agent, material agent, reasoning agent and negotiation agent.

The software agents were synthesised based on a new SiFA model presented in Chapter 3. Several agent characteristics such as autonomy, communication and conflict resolution ability were identified in Chapter 2. The following discussion presents an analysis of the software agents implemented in this research with respect to these characteristics. Table 6.12 shows the extent of autonomy achieved for agents in problem solving at various levels of granularity (from SiFAs to Multi-Agent). As the SiFAs combine to form Multi-Agents, the degree of autonomy increases from the control over the type and range of design variable to the solution of the design problem. However, the autonomy is also constrained by the nature of the given agent: the autonomy of the Geneva mechanism agent at the Multi-Agent level reflects its ability to solve a *domain-specific* problem solving (i.e. mechanism design) whereas the reasoning, negotiation and material agents exhibit autonomy for *relatively* generic problem solving.

Table 6.12 : Autonomy in software agents

Agent characteristic	Task agents	Levels of granularity in an agent			
		SiFA	SimFA	μFA	MA
Autonomy	Geneva mechanism design agent	- Autonomy assigned to constants and specified variables.	- The evaluation of task-specific functions (e.g., root stress).	- Control over wheel and driver perspectives. - Autonomy over individual utilities.	- Autonomy with respect to task-specific problem solving (e.g., Geneva mechanism design).
	Reasoning agent	- Autonomy of the agent controlled variables : nodes and their connectivity.	- Autonomous execution of functions such as construction of edges, determination of paths.	- Autonomy restricted to its functions: formation of network graph and determination of effective dependencies.	-
	Negotiation agent	- Determination of joint utilities.	- The evaluation of four individual game theory solutions.	- Restricted to a two player game	-
	Material agent	- Autonomy with respect to types of material properties	- Autonomous with respect to the evaluation of simple material property	– Interaction between simple material properties under specified conditions	-

Table 6.13 shows the communication ability achieved for each task agent. The

communication ability for the four agents follows a similar pattern over the SiFA-to-μFA transition. The communication mechanisms are implementation dependent. The agents use a shared class to communicate at the SiFA level and use KQML messages such as *ask* or *tell* at the level to communicate with other agents.

Table 6.13 : Communication in software agents

Agent characteristic	Task agents	Levels of granularity in an agent			
		SiFA	SimFA	μFA	MA
Communication with other agents	Geneva mechanism design agent	- Communication takes place via a shared class	- The variables are exchanged via Java message passing	- Through KQML messages but with limited performatives such as 'ask' or 'tell'	KQML message passing
	Reasoning agent				-
	Negotiation agent				-
	Material agent				-

Table 6.14 presents the ability of agents in the context of conflict resolution. The reasoning agent assists in the rationalisation of preferences whereas the negotiation agent reaches a final negotiated settlement for the conflicting agents.

Each agent in Table 6.14 shows certain level of reasoning and negotiation ability: from problem-dependent to generic. The former case involves simple rules [as in material agent] to support reasoning and detecting constraint violation at the SiFA level. The generic case involves procedures for determining qualitative and quantitative dependencies in a design network as seen at μFA stage in the reasoning agent. Game theory methods are applied for conflict negotiation at the μFA level where individual preferences between agents exist. At the Multi-Agent level, the Geneva mechanism agent possess the ability to detect internal conflicts.

Of the three characteristics considered in Tables [6.12 - 6.14], the main agent characteristic realised was conflict resolution ability. No known research prior to the author's work has reported an account of this agent characteristic realised in

Table 6.14 : Conflict resolution ability in software agents

Agent characteristic	Task agents	Levels of granularity in an agent			
		SiFA	SimFA	μFA	MA
Conflict resolution ability [Reasoning and Negotiation]	Geneva mechanism design agent	- Types of values for design variables a SiFA can assume - Range of values - Violation of value range for a SiFA resolved by relaxation	-Pre-specified domain equations - Constraint violation check - Conflict detection between design variables	- Partial design based on a specific perspective - Constraint violation detection within perspective - Consistent sub-design for its own perspective	- Prespective based design of Geneva mechanism (wheel and driver perspectives) - Identify conflicts between perspectives - Determine a negotiation range - Propagation of negotiated solutions to design variables
	Reasoning agent	- Vertex identification - Edge identification - Types of dependencies a SiFA can assume - Incorrect specification of vertices resolved via message passing	- Path identification - Path reduction - Effective qualitative and quantitative dependencies between desision and conflict variables	- Qualitative and quantitative reasoning for parametric design problems - Contributes to the rationalisation of agent utilities in a conflict resolution process	-
	Negotiation agent	- Joint utility - Range for joint utility	- Evaluation of a game theoretic solution (e.g., Nash solution) - Simple rule for isolating a solution based on equal pay-off	- Determines a single point solution for two-player games	-
	Material agent	- Types of values for material properties - Range of values for material properties	- Determine values of derived properties - Rules to identify properties for a given material	- Reply to a KQML message using appropriate performative	-

practical studies. This is a key enhancement from the understanding presented in previous research on collaborative approaches to product development [e.g., Cutkosky, *et al.*, (1996) and Balasubramanium and Norrie (1996)].

During the course of this case study other software agent characteristics such as adaptability and pro-activeness [as identified in Chapter 2] were not realised. Further research is required in realising these characteristics in the context of agent supported problem solving.

6.12.3 Other aspects of the collaborative process

New knowledge is gained in realising the complementary capabilities of collaboration technologies (software agents and CSCW approaches) in the context of conflict resolution. Such a realisation has not been achieved by known previous

research which often opted for binary solutions (*software agents* or *CSCW*). This case study of the design and manufacture of the Geneva mechanism provided a reinforcement of the earlier claim (in Chapter 5) of combining the two approaches of game theory and dependency based reasoning for resolving conflicts.

6.13 Summary

This chapter has been aimed at the validation of the two hypotheses in the author's research as identified in Chapter 1. This validation has been carried out through the case study of collaborative design and manufacture of a Geneva mechanism. In particular, the case study served as a test bed for examining decision related conflicts between agents. The combination of game theory and dependency based reasoning provided a useful mechanism to deal with conflicts arising in parametric design problems. The authors' research has been found to improve upon certain methodological limitations offered by the previous research with respect to conflict resolution between agents in collaborative product development process.

Furthermore, the case study on the design and manufacture the Geneva mechanism examined the resolution of conflicts involving software agents and humans in the broader context of a collaborative process. In particular, decision conflicts between (a) software - software agents (b) software agents - humans and (c) human - human designers/machinists due to variation in preferences were investigated in detail. The results obtained from this case study validated the author's agent synthesis and goal alignment hypotheses.

Chapter 7

Conclusions and future work

Trends in globalisation in industry have highlighted the need to support collaboration in distributed product development. This thesis has investigated the problem of conflict resolution in agent-supported collaborative design and manufacturing. This concluding chapter shows how the present research has achieved its objectives as identified in Chapter 1 and highlights its important contributions. Limitations of this work are also discussed and avenues for further research are identified.

To address the agent-supported collaboration, the problem of synthesising software agents was considered and a new agent model was proposed based on the idea of Single Function Agents (SiFA) and the Habermas's action theory. The transition of SiFAs to Multi-Agents was achieved systematically through increasing levels of complexity: the SiFAs, the Simple Function Agents (SimFAs), the Multi-Function Agents (μFAs) and Multi-Agents (MAs). At the μFA level, the agents have autonomy characterised by individual preferences for their goal. A difference between these preferences leads to conflicts between agents. This thesis therefore formalised conflicts between agents in terms of differing preferences represented as utility functions. A novel approach to conflict resolution was pro-

posed through a two-stage process consisting of dependency-based reasoning and goal alignment. The alignment of goals between agents was based on game-theoretic negotiation schemes. The proposed agent model and conflict resolution scheme were tested and validated through two engineering case studies: design of a poppet relief valve and design and manufacturing of the Geneva mechanism.

7.1 Conclusions

A detailed literature review conducted by the author led to the understanding that in the previous research on software agents for collaborative product development, little emphasis had been placed on:

- the synthesis of software agents to support collaboration;
- the conflict resolution between agents in a collaborative decision process.

Though software agent research has focused on issues from a computer science perspective (e.g., networking, interoperability), the agent synthesis with respect to application in engineering had not been addressed. In this sense, the software agent synthesis issues had largely been left unexplored. Although Berker and Brown (1996) proposed the idea of Single Function Agents (SiFAs) for agent synthesis, their SiFA model suffered from various drawbacks in terms of excessive complexity and inability to transform to Multi-Agents. In addition, Berker and Brown did not deal with the issue of conflict resolution nor did they apply their SiFA model to a practical engineering problem.

For conflict resolution between agents in the context of engineering, previous research offered two key techniques based on game theory and dependency based reasoning. These techniques were previously applied in isolation leading to the following drawbacks:

- The dependency based reasoning approach of Kusiak and Wang (1995) showed inability to deal with conflicts between agents due to differing preferences;
- Lack of mechanisms to ensure rationality in decision making as in the utility based approach of Kanappan and Marskek (1993);
- Lack of impartial mechanisms for mediation between conflicting agents.

The above considerations gave the motivation for new ideas which led to the proposal of the two hypotheses of this research outlined in Chapter 2, which, if satisfied, would improve upon some of the methodological limitations of the previous research in conflict resolution between agents.

This research has largely progressed along two related threads of investigation. The first thread focused on a model for agent synthesis to support collaboration. This was carried out by a new SiFA based model. The agents thus modelled had autonomy over their individual preferences which led to conflicts between them. The second thread of this research has focused on conflict resolution between agents for which a novel scheme was proposed. The rest of this section will address the achievements which have emerged from this investigation.

7.1.1 Agent synthesis

The author's first hypothesis (Agent synthesis) investigated the SiFA based approach for modelling complex software agents. A detailed analysis led to the identification of several drawbacks in the SiFA model of Berker and Brown (1996). These drawbacks were examined with respect to the following key characteristics:

- Number of agents, levels and types of agents;
- Use of SiFAs to form Multi-Agents (MAs).

The author's approach addressed these drawbacks by means of modelling agents based on their actions (e.g., communicative and discursive). The Habermas's theory of action was applied to software objects and a new SiFA model was proposed that had a compact structure.

As a result, in the new SiFA model, the number of decision levels was reduced to one, as compared to the unlimited number of decision levels as seen in the SiFA model of Berker and Brown (1996). The present approach systematically analysed how SiFAs can combine to form Multi-Agents. The author's approach took a novel view as full autonomy was not given to SiFAs at the parametric level. Through the transformation from SiFAs to Multi-Agents, there was an increase in the degree of autonomy of agents. The agents at the μFA level (representing design perspectives) had their own preferences which characterise their autonomy. These preferences were represented by utility functions. Resolution of Conflict due to different preferences was considered as a domain-independent task to be carried out via a novel scheme, as summarised next.

7.1.2 Conflict resolution between agents

The author's second hypothesis (Goal alignment) examined the resolution of conflicts between collaborating agents. This was carried out with a view that individual preferences do exist for agents in any collaborative work, a view that questions the cooperative assumption made in some previous research including that of Kusiak and Wang(1996) and Berker and Brown (1996).

The author's research has contributed towards the development of a novel conflict resolution scheme. This scheme has been aimed at providing alignment of preferences between collaborating agents while maintaining consistency in solutions. The novelty of the scheme lies in combining the key techniques for reasoning and negotiation:

- Dependency based reasoning;
- Game theory based negotiation.

The reasoning technique provided a means for maintaining consistency in design solutions. It utilised the domain equations for a design to define the constraints between decision and performance variables. Due their inter-dependency, changes in decision variables were propagated to all the related design variables via the associated design network. As a result of the reasoning process, a number of feasible designs were generated by collaborating agents while satisfying the given design constraints. However, each design variation had one or more conflict variables - the value of which was contested between the collaborating agents. This required the use of a negotiation scheme. The game-theoretic techniques for negotiation led to one or more single point solutions within the feasible space which were based on rational decision criteria of agent utilities. In addition, the scheme also provided certain mediation rules such as equal pay-offs for the conflicting agents (as in Kalai-Somordinsky solution), or maximising joint utility (as in Nash solution).

The conflict resolution scheme was bench-marked via a case study on the design of a poppet relief valve. Agent-supported design of a poppet relief valve was considered and conflict was identified between the differing preferences of two design perspectives: the spring and the enclosure. A quantitative comparison of negotiation results with those of Kusiak and Wang (1996) and Kanappan and Marshek (1993) was presented. The results obtained showed the applicability of the new scheme for resolving conflicts between software agents. In addition, the results reinforced the argument that conflict resolution required both dependency-based reasoning (to determine a feasible design space) and negotiation (to allow agents to rationally agree on a single feasible solution).

The case study of the design and manufacture of a Geneva mechanism was carried out to validate the conflict resolution scheme in a broader context of a collab-

orative design and manufacturing process involving both software agents and humans. In particular, this case study examined the conflicts at various stages of product development, arising due to differing preferences between (a) software - software agents (b) software agents - human designers and (c) human designers - human machinists.

The following conclusions can be drawn from the Geneva mechanism case study:

- An engineering product development process involves a large number of design variables, but a small number of controlling decision variables and performance variables;
- A collaborative decision process involves a number of agents, each agent representing specific design perspective;
- Design agents representing specific perspectives can be synthesised using the proposed SiFA model;
- The dependency based technique can be effectively used to guide rationality in design decisions;
- Each agent has its own preferences involving controlling decision variables;
- The utility functions can adequately express an agent's preferences for decision and performance variables;
- A feasible design space can be determined from the dependency network and initial specification;
- Single point solutions can be obtained within the feasible space by using game theoretic techniques. This ensures the alignment of goals between agents;
- The mediation rules can be used in isolating a final negotiated solution and thereby avoiding secondary conflicts.

Having presented the general conclusions which highlighted the main contributions of the author's research, its implications in a wider industrial context are explored next.

7.2 The implication of this research

The significance of this research can be understood by considering its relevance to industrial trends. The relevance is examined along the two inter-related threads upon which the author chose to explore this research.

The research has demonstrated in a laboratory-based controlled environment how software agents could effectively address the issue of conflict resolution. The author addressed conflict resolution between agents in a collaborative product development process. Companies such as Enviros systems have used reasoning techniques based on qualitative dependencies to develop tools for decision making based on Multiple Criteria Decision Making (MCDM) in the context of chemical engineering processes [Brice and Johns (1999)]. It could employ the proposed conflict resolution schemes in the presence of competing objectives. Another effort by IBM T J Watson research centre presented decision making using software agents to minimise waste and transportation costs [Price (1999)].

Though the above efforts acknowledge the existence of multiple goals in their decision processes, they treated decision making largely as a problem of optimisation rather than of negotiation. The agents in a decision process have autonomy over their individual preferences towards a common goal. As agents collaborate, conflicts often occur due to difference in these preferences. Depending on the situation, the agents may need to generate dynamic preferences characterised by their adaptability. The qualities such as autonomy and adaptability complement the agent's ability to generate preferences and identify their goals in a conflict situation. Given sufficient enhancements in software agent development and synthesis

(in the context of autonomy and adaptability), these decision making procedures could make use of conflict resolution scheme such as the one developed by the author for dealing with conflicts between agents in a distributed collaborative process.

To this end the discussion highlighted the possible areas where author's research efforts could be utilised in practical applications. However, there still exist several limitations in this thesis, as identified below.

7.2.1 Limitations of this research

This research uncovered several key issues relating to agent supported collaborative product development. In particular, there are several issues related to scalability and generalisation of the proposed technique that need to be addressed.

- Though the synthesis of software agents was considered in this research, it was not possible to model all their behavioural characteristics. The main reason for this was that the primary research emphasis was on the conflict resolution between software agents;
- Only conflicts between two agents were considered in the present study. To deal with the multiple-agent negotiation (agents > 2), the present conflict resolution scheme will be inadequate. The scalability of the conflict resolution scheme also needs to be considered;
- The agent synthesis model presented a compact way of constructing multi-agents for engineering problem solving. However, the application of the SiFA model to large engineering products such as aircrafts is yet to be established;
- The agents presented in this study do not have learning ability or adaptability and are thus static. Such abilities would complement the agent's

behaviour in conflict resolution by guiding it to dynamically change its preferences;

- The representation of preferences is quantitative and deterministic in the present case which need not always be true. The agent synthesis model will therefore need to be scalable to incorporate a broad range preferences that are quantitative as well as fuzzy in nature. Such a model would address a broad range of conflicts from conceptual to detailed design and manufacturing stages;

- For large scale problems which involve technical as well as economic considerations, the applicability of the agent synthesis model and the conflict resolution scheme would need further investigation;

Finally, real industrial problems will present a much more complex design process than the ones considered here. It is, however, acknowledged that a laboratory-based validation approach is central to all scientific and engineering research, and serves as an essential step towards dealing with more complex real-life applications [Johanssen, *et al.*, (1994), Ishii, *et al.*, (1994)]. The industrial cases will typically involve several agents, additional constraints such as cost and time, which were not considered in this study. Further research is required in respect of such dynamic constraints. Further validation of the proposed methodologies for agent synthesis and conflict resolution is necessary by undertaking industrial case studies.

7.3 Future work

This research has given rise to challenging issues that are worth exploring further. Conflict resolution between agents as seen in this thesis is important for agent supported collaborative product development. However, real world design

problems would simultaneously involve several agents. The problem of conflicts between n-agents has not been previously investigated as a core research issue except for some cursory remarks on coalition building [Rao (1999)].

Investigation into n-agent negotiation would be of immense value but it would require dynamics characterisation of their individual preferences, adequate mathematical modelling including collectively stable equilibrium, uniqueness and geometrical representation of solutions. Techniques for defining multi-agent joint utility functions will also need to be developed.

In engineering problems there are situations where exact domain equations cannot be evaluated and only fuzzy or approximate characterisations can be made [Wang (1999)]. Thus there is a need to extend the reasoning theory particularly in the direction of fuzzy and approximate representations using techniques such as response surface methodology [Hacker and Lewis (1998)]. Such an extension would support generic conflict resolution while maintaining rationality in negotiated solutions.

7.4 Final overview

The author's research has focused on agent-supported conflict resolution in collaborative product development. It presented a review of collaboration technologies and isolated certain deficiencies in previous approaches in the context of agent supported conflict resolution. A new agent model based on Habermas's action theory was used to build software agents for the purpose of the case studies. The resolution of conflicts between agents was further investigated via a novel conflict resolution scheme. This scheme was developed to resolve parametric conflicts between agents while maintaining rationality and satisfy design constraints.

Two case studies were carried out to examine the validity of the new agent model

and the new conflict resolution scheme in the context of collaborative product development. Based on the results and conclusions from these case studies, it was found that the author's thesis significantly furthers the state-of-the-art in conflict resolution using agents in collaborative product development.

Published papers

Sreeram, R. T. and P. K. Chawdhry, (1999), "Human-centred Integration of a Stand-alone Manufacturing Facility in a Networked Product Development Environment", *International Journal of Computer Integrated Manufacturing*, Vol. 12, No.4, pp. 338-360.

Sreeram, R. T. and P. K. Chawdhry, (1999), "Towards the Unification of Game-Theoretic and Constraint Relaxation Schemes", 6^{th} *ISPE Conference on Concurrent Engineering: Research and Applications*, Bath, UK.

Sreeram, R. T. and P. K. Chawdhry, (1999), "A Unified Scheme for Conflict Negotiation in a Multi-Agent Decision Process", 6^{th} *ISPE Conference on Concurrent Engineering: Research and Applications*, Bath, UK.

Sreeram, R. T. and P. K. Chawdhry, (1998), "A Management Case Study of an Agile Virtual Enterprise", *International Journal of Agile Manufacturing*, Vol. 1, No.2, pp. 201-212.

Sreeram, R. T. and P. K. Chawdhry, (1998), "A Single Function Agent Framework for Task Decomposition and Conflict Negotiation", *ASME Design Engineering Technical Conferences, Design for Manufacturing Conference*, DFM-5748, Atlanta, Georgia, USA.

Sreeram, R. T. and P. K. Chawdhry, (1998), "Integration of Collaborative Ap-

proaches for Distributed Product Development", In: *Globalisation of Manufacturing in the Digital Communications Era of the 21^{s}t Century: Innovation, Agility and the Virtual Enterprise*, (eds.) Gianni, J., Gustav, J. O., Preiss, K., and Wozny, M., pp. 149-162.

Sreeram, R. T. and P. K. Chawdhry, (1998), "A Game Theoretic Approach to Conflict Negotiation in a Single Function Agent Environment", 5th *ISPE Conference on Concurrent Engineering: Research and Applications* Tokyo, Japan. pp. 515-524.

Sreeram, R. T. and P. K. Chawdhry, (1998), "A Design Reuse Model for Collaborative Product Development", *Engineering Design Conference*, Brunel, London, pp. 155-162.

Sreeram, R. T. and P. K. Chawdhry, (1997), "A Java Agent for Communication in a Concurrent Distributed Product Development Environment", 4th *ISPE Conference on Concurrent Engineering: Research and Applications* Detroit, USA. pp. 159-166.

References

Ainslie, P. J., Knight, G. D., & Schauerte, F. J. 1985. Integrated Circuit Designing. *Automotive Engineering*, **93**, 40–47.

Akao, Y. 1990. *Quality Function Deployment,*. Productivity Press, MA, USA.

Alford, H. 1994. Cellular Manufacturing - The Development of the Idea and its Application. *New Technology, Work and Employment*, **9**(1), 3–17.

Arrow, K. J. 1964. *Social Choice and Industrial Values.* John Wiley and Sons, New York.

Badrinath, K., & Rao, J. R. J. 1996. Modelling for Concurrent Design Using Game Theory. *CERA journal*, **4**, 389–401.

Balasubramanium, S., & Norrie, D. H. 1996. A Multiagent Architecture for Concurrent Design. *CERA journal*, **4**, 7–16.

Bentley, R., Rodden, T., Sawyer, P., & Sommerville, I. 1994. Architectural Support for Cooperative Multiuser Interfaces. *IEEE Computer*, **27**, 37–46.

Berker, I., & Brown, D. C. 1996. Conflicts and Negotiation in Single Function Agent Based Design Systems. *CERA journal*, **4**(1), 17–33.

Binmore, K. G. 1985. *Bargaining and Coalitions.* Cambridge University press, United Kingdom. Chap. Game-Theoretic Models of Bargaining.

Boden, A. M. 1994. The Role of Emotion in Believable Agents. *Communications of the ACM*, **37**, 117–121.

Boothroyd, G. 1988. Development of DFMA and its Impact on US Industry. *In: CE-99.*

Brice, A., & Johns, B. 1999. *Improving Design by Improving the Design Process.* Tech. rept. QSL-900-2A-WP-001. Enviros Software Solutions, Oxfordshire, UK.

Brown, D. C. 1992. *Encyclopedia of AI.* John Wiley, New York, USA.

Brynes, J. 1993. The Virtual Corporation. *Business Week*, 98–104.

Cutkosky, M. R., Tenenbaum, J. M., & Glicksman, J. 1996. Madefast:Collaborative Engineering over the Internet. *Communications of the ACM*, **39**, 78–87.

Davey, A., & Olson, D. 1998. Multiple Criteria Decision Making Models in Group Decision Support. *Group Decision and Negotiation*, 55–75.

Deitz, D. 1997. *Product Development on the Web.* Mechanical Engineering.

deKleer, J., & Brown, J. S. 1984. A Qualitative Physics Based on Confluences. *Artificial Intelligence*, **24**(1-3), 7–84.

Dunskus, B. V., Grecu, B. V., Brown, D. C., & Berker, I. 1995. Using Single Function Agents to Investigate Conflict. *Aiedam Journal*, **9**, 299–312.

Durfee, E. H., Kiskis, D. L, & Birmingham, W. P. 1997. The Agent Architecture of the University of Michigan Digital Library. *IEE proceedings in Software Engineering*, **144**(1), 61–71.

Erkes, J. W., Kenny, K. B., Lewis, J. W., Sarachan, B. D., Sobolewski, M. W., & Sun, R. N. 1996. Implementing Shared Manufacturing Services on the World-Wide Web. *Communications of the ACM*, **37**, 34–45.

Etzioni, O., & Weld, D. 1994. A Softbot-based Interface to the Internet. *Communications of the ACM*, **37**, 72–76.

Finin, T., Labrou, Y., & Mayfield, J. 1995. *KQML as an Agent Communication Language.* Tech. rept. Department of Comuter Science, University of Maryland, Baltimore, USA.

Frost, H. R., & Cutkosky, M. R. 1996 (August 18-22). Design for Manufacturability via Agent Interaction. *In: ASME Design Engineering Technical Conferences, Design for Manufacturing Conference, irvine, californis, usa.*, vol. 96-DETC, DFM-1302.

Gasser, L. G. 1988. *Readings in Distributed Artificial Intelligence.* Morgan Kaufmann Publishers, New York.

Gay, G., & Lentini, M. 1995. *Use of Communication Resources in a Networked Collaborative Design Environment.* Tech. rept. Interactive Multimedia Group, Cornell University, 209 Kennedy Hall, Ithaca, NY 14853-4203, USA.

Genesereth, M. R., & Ketchpel, S. P. 1994. Software Agents. *Communications of the ACM*, **37**, 48–53.

Gonikhin, O., & Medland, A. J. 1990. The Use of Networks in the Description of the Design to Manufacturing Process. *International Journal of Computer-Integrated Management Systems*, **3**(3), 171–177.

Grudin, J. 1994. CSCW: History and Focus. *IEEE Computer*, **27**, 19–26.

Habermas, J. 1984. *Theory of Communicative Action: Reason and Rationalisation of Society.* Beacon Press, Boston, USA.

Hacker, K., & Lewis, K. 1998. Using Robust Design Techniques to Model the Effects of Multiple Decision Makers in a Design Process. *In: ASME Design Engineering Technical Conferences, Design Automation Conference*, vol. DAC-5604. Atlanta, Georgia, USA.

Hanson, B. A. 1984. *Independent Craftsmen and the Labour Process: A Study of Continuity and Change in Craft Production from Medieval to Modern Times.* PhD Thesis, University of Bath, UK.

Hardwick, M., Spooner, D. L., Rando, T., & Morris, K. C. 1996. Sharing Manufacturing Information in Virtual Enterprises. *Communications of the ACM*, **39**, 34–45.

Hasty, C. E., & Potts, J. F. 1966. Analysis and Synthesis Procedures for Geneva Mechanism Design. *IBM Journal of Research and Development*, **10**(3), 186–197.

Hill, S. A. 1999. *Private Communication.* NASA Langley Research Centre, Virginia 23681, USA.

Hirschheim, R. A. 1985. *Office Automation Concepts, Technologies and Issues.* Addison-Wesley Publishing Company, New York.

Huhns, M. N., & Singh, M. P. 1994. Automating Workflows for Service Provisioning - Integrating AI and Database Technologies. *In: Proceedings of IEEE Conference of Artificial Intelligence for Applications.*

Ishii, H., Kobayashi, M., & Arita, K. 1994. Interactive Design of Seamless Collaboration Media. *Communications of the ACM*, **37**, 83–97.

Jelassi, M. T., & Foroughi, A. 1989. Negotiation Support Systems: An Overview of Design Issues and Existing Software. *Decision Support Systems*, **5**, 167–181.

Johanssen, G., Levis, A. H., & Henk, G. S. 1994. Theoretical Problems in Man-Machine Systems and Their Validation. *Automatica*, **30**, 11–25.

Johnson, R. C. 1956. How to Design Geneva Mechanisms to Minimize Contact Stress and Torsional Vibration. *Machine Design*, **28**(6), 107–111.

Kalai, E., & Somordinsky, M. 1975. Other Solutions to Nash's Bargaining Problem. *Econometrica*, **43**(3), 513–518.

Kannapan, S. M., & Marshek, K. M. 1993. *Concurrent Engineering: Automation, Tools, and Techniques.* John Wiley Sons, Inc. New York, USA. Chap.

An Approach to Parametric Machine Design and Negotiation in Concurrent Engineering.

Kersten, G. K. 1988. A Procedure for Negotiating Efficient and Non-efficient Compromises. *Group Decision and Negotiation*, **2**, 259–278.

Klein, M. 1990. *Conflict Resolution in Cooperative Design.* Ph.D. thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, USA.

Klein, M. 1993. Supporting Conflict Management in Co-operative Design Teams. *Group Decision and Negotiation*, **2**, 259–278.

Klein, M., & Lu, S. C. Y. 1995. Conflict Resolution in Co-operative Design. *Artificial Intelligence in Engineering*, **4**(4), 168–180.

Kuipers, B. 1986. Qualitative Simulation. *Artificial Intelligence*, **29**, 289–338.

Kusiak, A., & Wang, J. 1995. Dependency Analysis in Constraint Negotiation. *IEEE Transactions on Systems, Man and Cybernetics*, **25**(9), 1301–1313.

Kusiak, A., Wang, J., & He, D. W. 1996. Negotiation in Constraint-based design. *ASME Journal of Mechanical Design*, **188**, 470–477.

Lander, S. E., & Corkill, D. D. 1996. Designing Integrated Engineering Environments: Blackboard Based Integration of Design and Analysis Tools. *CERA Journal*, **4**(1), 59–71.

Lee, T. W. 1981. Optimizations of High Speed Geneva Mechanisms. *ASME Journal of Mechanical Design*, **103**, 621–630.

Lewis, A., Cosier, G., & Nightingale, C. 1999. Whither Video? Videotelephony in Perspective. *British Telecommunications Engineering*, **17**, 245–250.

Lewis, K., & Mistree, F. 1998. Collaborative, Sequential and Isolated Decisions in Design. *ASME Journal of Mechanical Design*, **120**, 643–652.

Lyons, J. L. 1982. *Valve Designer's Handbook.* Van Nostrand Reinhold, New York.

M., Gere J., & Timoshenko, S. P. 1991. *Mechanics of Materials.* Chapman & Hall London.

March, J. G., & Simon, H. A. 1993. *Organisations.* Blackwell, Oxford, UK.

McGuire, J. G., Kuokka, D. R., Weber, J. C., Tenenbarm, J. M., Gruber, T. R., & Olsen, G. R. 1993. SHADE: Technology for Knowledge-based Collaborative Engineering. *CERA Journal*, **1**(3), 137–147.

Medland, A. J. 1992. *The Computer-based Design Process.* Chapman and Hall, London, UK.

Meier, R. L., Walker, H. F., & Wallingord, E. 1994. Agile Manufacturing: Production Systems, Organizational Elements and Performance Measurements. *Flexible Automation and Integrated Manufacturing*, 290–299.

Nash, J. 1950. The Bargaining Problem. *Econometrica*, **18**(2), 155–162.

Ndumu, D. T., & Nwana, H. S. 1996. An Introduction to Agent Technology. *BT Technology Journal*, **14**, 55–67.

Ndumu, D. T., & Nwana, H. S. 1997. Research and Development Challenges for Agent-based Systems. *IEE Proceedings on Software Engineering*, **144**, 38–50.

Ngwenyama, O. K., & Lyytinen, K. J. 1997. Groupware Environments as Action Constitutive Resources: A Social Action Framework Analysing Groupware Technologies. *Computer Supported Cooperative Work: The Journal of Collaborative Computing*, **6**, 71–93.

Nishiguichi, T. 1994. *Strategic Industrial Outsourcing: The Japanese Advantage.* Oxford University Press, New York.

Obrst, L., Woytowitz, M., Rock, D., Lander, S., Gallagher, K., & Decker, K. 1997. Agent-Based Integrated Product Teams. *In: ASME Design Engineering Technical Conferences.* California, USA.

Oh, V., & Sharpe, J. 1995. Conflict Management in an Interdisciplinary Design Environment. *AIEDAM Journal*, **9**, 247–258.

Oliver, J. R. 1996. *On Artificial Agents for Negotiation in Electronic Commerce.* PhD Thesis, Wharton School of Management, University of Pennsylvania, Philadelphia, USA.

Pahl, G., & Beitz, W. 1996. *Engineering Design: A Systematic Approach.* Springer, New York.

Petrie, C. J. 1996. Agent-based Engineering, the Web, and Intelligence. *IEEE Expert*, **12**, 24–29.

Poolton, J., & Barclay, I. 1996. Concurrent Engineering Assessment: A Proposed Framework. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, **10**(4), 321–328.

Price, D. 1999. Intelligent Agents Trim Paper Making Costs. *IEEE Intelligent Systems*, **14**(2), 4–5.

Rahim, M. A. 1986. *Managing Conflict in Organisations.* Praegar, New York, USA.

Raiffa, H. 1972. *The Art and Science of Negotiation.* Harvard University Press, Massachussets, USA.

Rangaswamy, A., & Shell, G. R. 1994. Using Computers to Realize Joint Gains in Negotiations: Toward an 'Electronic Bargaining Table'. *Management science*, **43**(8), 1147–1163.

Rao, J. R. J. 1999. *Private Communication.* Department of Mechanical Engineering, University of Houston, Texas 77204, USA.

Rao, J. R. J., Badrinath, K., Pakala, R., & Mistree, F. 1997. A Study of Optimal Design under Conflict using Models of Multi-player Games. *Engineering Optimisation*, **28**, 63–94.

Rao, S. S. 1987. Game Theory Approach for Multi-objective Structural Optimisation. *Computers and Structures*, **25**, 119–127.

Rao, S. S., & Freheit, T. I. 1991. A Modified Game Theory Approach to Multi-objective Optimisation. *ASME Journal of Mechanical Design*, **133**, 286–291.

Reinhard, W., Schweitzer, J., Volksen, G., & Weber, M. 1994. CSCW Tools: Concepts and Architectures. *IEEE Computer*, **27**, 28–36.

Rothkopf, M. H., & Harstad, R. M. 1972. Modelling Competitive Bidding: A Critical Essay. *Management Science*, **40**(3), 364–384.

Sandholm, T. W., & Lesser, V. R. 1998. *Coalitions among Computationally Bounded Agents.* Tech. rept. Washington University, Department of Computer Science, St. Louis, MO, USA.

Schiffner, N. 1998. Distributed 3D Virtual Environment for Collaborative Engineering. *In: Proceedings of the Tenth International IFIP WG5.2/5.3 Conference.* Trento, Italy.

Shakeri, C. 1998. *Discovery of Design Methodologies for the Integration of Multidisciplinary Design Problems.* Ph.D. thesis, Department of Mechanical Engineering, Worcester Polytechnic Institute, USA.

Shardlow, N. 1990. *Action and Agency in Cognitive Science.* Master's Thesis, Department of Psychology, University of Manchester, Manchester M13 9PL, UK.

Shoham, Y. 1999. What we Talk about when we Talk about Software Agents. *IEEE Intelligent systems*, 28–31.

Smith, R. P., & Eppinger, S. D. 1997. A Predictive Model of Sequential Iteration in Engineering Design. *Management Science*, **43**(8), 1104–1120.

Sobek, D. K., Ward, A. C., & Liker, J. K. 1999. Toyota's Principles of Set-based Concurrent Engineering. *Sloan Management Review*, **Winter issue**, 67–83.

Sobolewski, M. W., & Erkes, J. W. 1995. CAMnet: Architecture and Applications. *In: CE-95.* McLean, Virginia, USA.

Soo, V. W., & Wang, T. C. 1992. Integration of Qualitative and Quantitative Reasoning in Iterative Parametric Mechanical Design. *AIEDAM Journal*, **6**(2), 95–109.

Sycara, K., & Lewis, C. M. 1991. Modelling Group Decision Making and Negotiation in Concurrent Product Design. *International Journal of Systems Automation: Research and Applications*, **1**, 217–238.

Takeda, K., Inaba, M., & Sugihara, K. 1996. User Interface and Agent Prototyping for Flexible Working. *IEEE Expert*, **3**(2), 40–50.

Theobald, G. 1995. *Modelling Standard Components for the Design and Optimisation of Engineering Systems.* PhD Thesis, University of Bath, UK.

Thomson, W., & Lensberg, T. 1989. *Axiomatic Theory of Bargaining with Variable Number of Agents.* Cambridge University Press, New York, USA.

Toye, G., & Tenenbaum, J. M. 1993. SHARE: A Methodology and Environment for Collaborative Product Development. *IEEE Infrastructure for Collaborative Enterprises*, 1–16.

Victor, S. K., Brown, D. C., Bausch, J. J., Zenger, D. C., Ludwig, R., & Sisson, R. D. 1993. Using Multiple Expert Systems with Distinct Roles in a Concurrent Engineering System for Powder Ceramic Components. *Applications of Artificial Intelligence in Engineering*, **1**, 83–96.

Vincent, T., & Grantham, W. 1981. *Optimality in Parametric Systems.* John Wiley Sons, New York, USA.

Walklet, R. H. 1989. Continuos Improvement and Simultaneous Engineering. *Automative Engineering*, **97**(10), 59–63.

Wang, J. 1994. *Analysis of Processes and Constraints in Engineering Design.* Ph.D. thesis, Department of Industrial Engineering, University of Iowa, USA.

Wang, J. 1999. *Private Communication.* Department of Industrial Engineering, Feng Chia University, Taiwan.

Ward, A., Liker, J., Cristiano, J. L., & Sobek II, D. K. 1995. The Second Toyota Paradox: How Delaying Decisions Can Make Better Cars. *Sloan Management Review*, **36**(3), 43–61.

Waterson, P. E., Clegg, C. W., Bolden, R., Pepper, K., Warr, P. B., & Wall, T. D. 1999. The Use and Effectiveness of Modern Manufacturing Practices: A Survey of UK Industry. *International Journal of Production Research*, **37**(10), 2271–2292.

Weiss, M. A. 1998. *Data Structures & Problem Solving Using Java.* Addison-Wesley, Harlow, England.

Wellman, M. P. 1990. *Formulation of Tradeoffs in Planning under Uncertainity.* Pitman Publishing, London, United Kingdom.

Womack, J. P., Jones, D. T., & Roos, D. 1991. *The Machine that Changed the World: The Story of Lean Production.* Harper Perennial, New York.

Wooldridge, M., & Jennings, N. R. 1995. Intelligent Agents: Theory and Practice. *Knowledge Engineering Review*, **10**(2), 115–152.

Yang, F. C., & Ho, Y. K. 1994 (August 29-31). A Framework for Cooperative Distributed Problem Solving for Office Automation. *In: CE-94, mclean, virginia, usa.*

Yeaple, F. 1979. Geneva Mechanism for Indexing. *Product Engineering*, **50**(8), 39–43.

Zeng, D., & Sycara, K. 1998. Bayesian Learning in Negotiation. *International Journal of Human-Computer Studies*, **48**, 125–141.

Appendix A

Additional details on the Poppet relief valve design case study

This appendix provides additional details for the Poppet valve design case study.

A.1 Poppet valve design

A.1.1 Nomenclature

$A(m^2)$ = Pipeline cross-section area

$A_1(m)$ = Corrosion resistance allowance for valve enclosure

$A_2(m)$ = Corrosion resistance allowance for poppet valve stem

A_{cl} = Clash allowance ratio

C = Helical spring index

C_v = Valve configuration factor

$C_l^1(m)$ = Radial clearance between helical spring and enclosure

$C_l^2(m)$ = Radial clearance between helical spring and poppet stem

C_f = Orifice coefficient

$d(m)$ = Helical spring wire diameter

$d_h(m)$ = Valve hole diameter

$d_{eo}(m)$ = Equivalent orifice diameter

$d_o(m)$ = Orifice diameter

$d_s(m)$ = Seal diameter

$d_L(m)$ = Flow line diameter

$D(m)$ = Mean helical spring diameter

$D_i(m)$ = Inner diameter of spring enclosure

$D_o(m)$ = Outer diameter of spring enclosure

$D_v(m)$ = Valve outer diameter

$F_c(N)$ = Cracking force on helical spring

$F_d(N)$ = Dynamic fluid force

$F_t(N)$ = Total force on helical spring

$g(m/sec^2)$ = Acceleration due to gravity

$G(GPa)$ = Shear modulus of spring material

$K_s(N/m)$ = Computed spring rate

$K_{act}(N/m)$ = Actual spring rate

K = Flow coefficient (head loss factor)

K_w = Wahl spring factor

$L_f(m)$ = Helical spring free length

$L_i(m)$ = Helical spring installed length

$L_s(m)$ = Helical spring solid length

n = Factor of safety

N = Number of helical spring coils

P (N/mm^2) = Maximum fluid pressure

P_{matl} = Material for Valve enclosure

$P_c(N/mm^2)$ = Cracking pressure

(N/mm^2) = Pressure drop

$Q(cu.m/sec)$ = Fluid flow rate

r_1 = Flow area variation ratio

r_2 = Spring rate reduction ratio

r_c^1 = Spring outer diametral clearance ratio

r_c^2 = Spring inner diametral clearance ratio

r_h = Valve hole size ratio

S = Fluid specific gravity

S_{matl} = Helical spring material

$S_p(N/mm^2)$ = Allowable stress for the pipe

$t_p(m)$ = pipe thickness

$t_s(m)$ = seal thickness

$t_v(m)$ = Valve thickness

$\delta_{max}(m)$ = Maximum deflection

$\rho(Kg/m^3)$ = Density of water (62.4)

$\tau_{min}(N/mm^2)$ = Helical spring minimum stress

$\tau_{all}(N/mm^2)$ = Allowable stress for spring

Design equations

$$d_0 = \frac{Q}{29.81C_f}^{0.5} \frac{S}{\Delta P}^{0.25}$$

$d_{e0} = d_0$

$$d_{e0} = C_v(d_L)^{1.07}$$

$$d_{e0} = \frac{1.292 d_L}{\sqrt[4]{K}}$$

$$A = \frac{\pi d_L^2}{4}$$

$$t_s = d_s - d_L$$

$$F_c = P_c \frac{\pi d_s^2}{4}$$

$$F_d = \frac{0.0007 Q^2 S \rho_w}{gA}$$

$$\delta = r_1 \frac{d_L}{4}$$

$$k_s = \frac{F_s}{\delta}$$

$$F_t = F_c + F_d$$

$$r_2 = \frac{K_{act}}{K_s}$$

$$K_w = \frac{4C-1}{4C-4} + \frac{0.615}{C}$$

$$\tau = K_w \frac{2.55 F_t C}{d^2}$$

$$\tau_s = \frac{\tau_{all}}{n}$$

$$C = \frac{D}{d}$$

$$N = \frac{Gd^4}{8D^3 K_{act}}$$

$$C_{l1} = D r_{c1}$$

$$C_{l2} = D r_{c2}$$

$$L_s = d(N+2)$$

$$L_f = L_i + \frac{F_c}{K_{act}}$$

$$D_i = D - \frac{d}{2} - C_{l2}$$

$$D_o = D + \frac{d}{2} + C_{l1}$$

$$L_i = L_s + \delta + A_{cl}\delta$$

$$t_p = A_1 + \frac{PD_v}{2S_p}$$

$$t_v = A_2 + \frac{PD_i}{2S_p}$$

$$D_o = D_v - t_v$$

$$d_h = D_i - t_v$$

A.1.2 Poppet valve design agent details

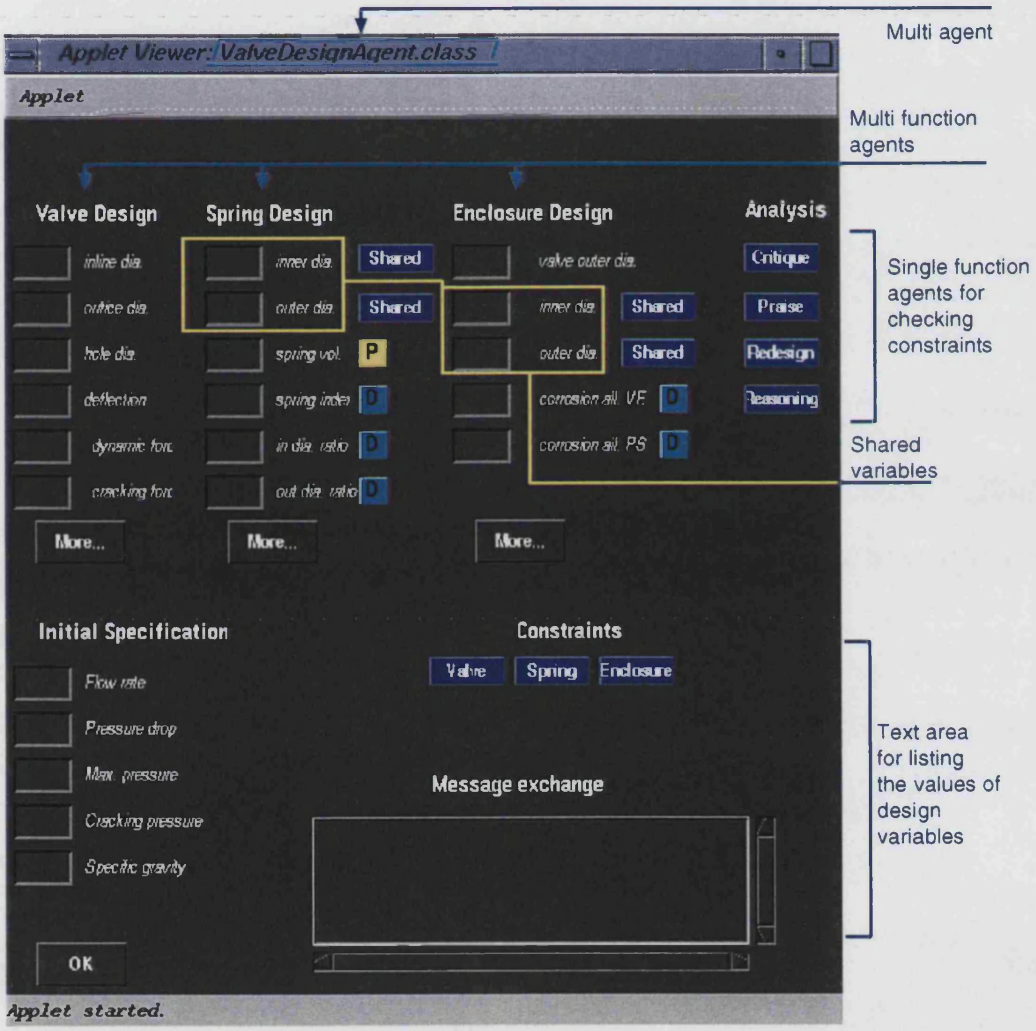


Figure A.1: GUI of the Poppet relief valve agent

Appendix B

Additional details on the Geneva mechanism design and manufacturing case study

B.1 Geneva mechanism design

B.1.1 Nomenclature

β_0 = Angle of Geneva wheel (deg.)

R = Radius of Geneva wheel (mm)

D = Diameter of Geneva wheel (mm)

r_L = Locking drum radius (mm)

a_{max} = Maximum acceleration of geneva wheel (rad/sec^2)

S^w = Wheel slot width (mm)

w^{tip} = Wheel tip width (mm)

w^t = Wheel thickness (mm)

N_s = Number of slots

C = Distance between centres of wheel and driver (mm)

S = Wheel slot distance (mm)

I_L = Load inertia (kg/mm^4)

I_G = Moment of inertia of geneva wheel (kg/mm^4)

MAT_{wheel} = Material of the wheel

ρ_{wheel} = Density of wheel material (kg/mm^3)

ν_{wheel} = Poisson's ratio of wheel material

E_{wheel} = Modulus of elasticity of pin material (Pa)

σ_T = Tip stress (Pa)

σ_R = Root stress (Pa)

C_L = Clearance between wheel and slot (mm)

K_1,K_2,K_3,K_4 and K_5 = design constants

k_1^s,k_2^s = Stress concentration factors

μ = Coefficient of friction

α_0 = Angle of driver (deg.)

γ = Wheel locking angle (deg.)

r_P = Pin radius (mm)

r_D = Driver radius (mm)

σ_C = Pin contact stress (Pa)

MAT_{pin} = Material of the wheel

ρ_{pin} = Density of wheel material (kg/mm^3)

ν_{pin} = Poisson's ratio of wheel material

E_{pin} = Modulus of elasticity of pin material (Pa)

Design equations

$$\beta_0 = \frac{\pi}{N_s}$$

$$S = C(1 - \sin\beta_0)$$

$$D = 2C\cos\beta_0$$

$$r_L = r_1 - \frac{d}{2} - w_{tip}$$

$$w_{slot} \geq 2r_p$$

$$w_T = K_3 D$$

$$I_g = I - N_s I_1 - 2N_s I_2$$

$$\sigma_T = \frac{6P_d X}{W h_t^2}$$

$$y = \frac{D}{2} - \rho + X$$

$$\sigma_R = \frac{\frac{6K_1 P_d(l_1 + \mu l_2)}{W h_R'^2}}{\frac{K_2 P_d(\sin\beta_0 + \mu\cos\beta_0)}{W h_R'}}$$

$$h_R' = 2(S - 0.01)\sin\beta_0 - 2r_p$$

$$l_1 = \rho - (\cos\beta_0)^2(S - 0.01)$$

$$l_2 = (S - 0.01)\cos\beta_0 \sin\beta_0 - \frac{2r_p}{2}$$

$$C_L \approx 0.0254D$$

$$P_{max} = \frac{(I_g + I_l)a_{max}}{\rho_{max}}$$

$$\rho_{max} = K_1 D$$

$$r_p = K_2 D$$

$$\nu = \frac{\pi(n+2)}{n}$$

$$\alpha_0 = \frac{\pi}{2}\frac{(n-2)}{n}$$

$$r_d = C \sin \beta_0$$

$$\sigma_C = 3237\sqrt{\frac{P_{max}}{w_T 2 r_p}}$$

B.1.2 Geneva mechanism design agent details

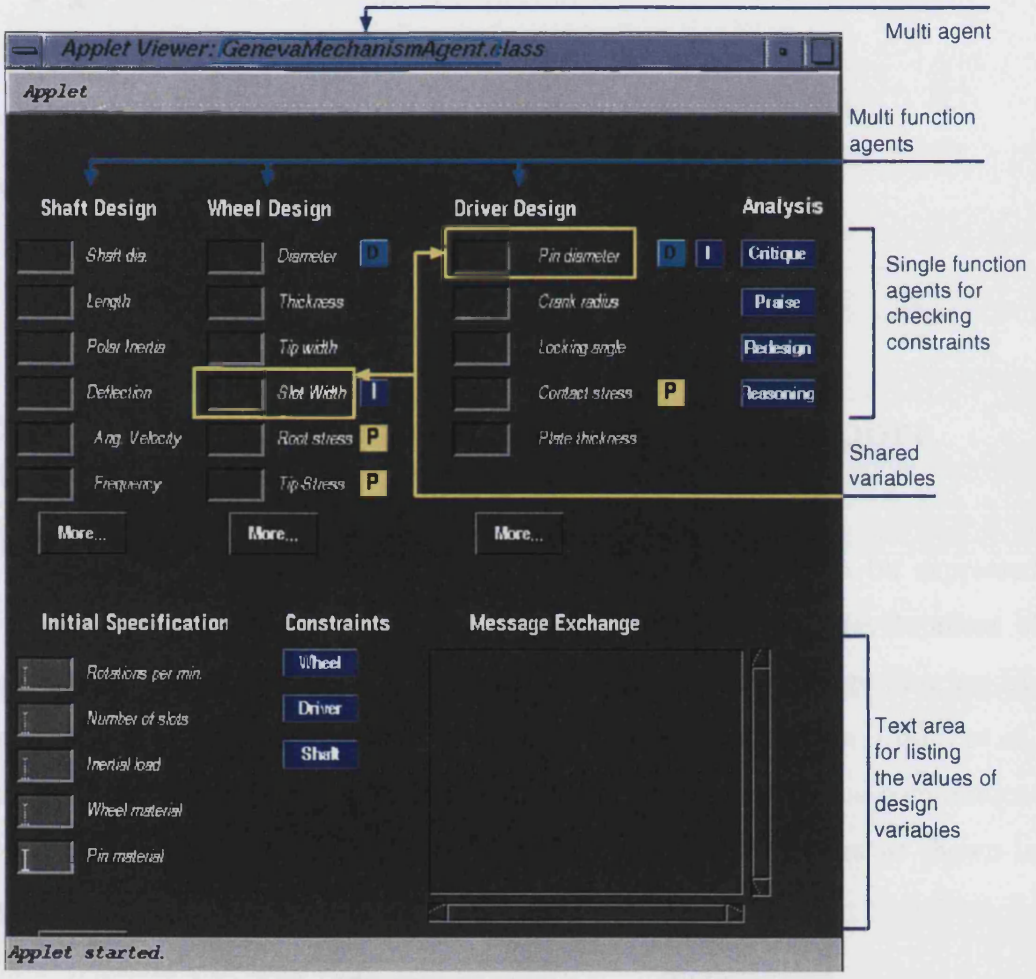


Figure B.1: GUI of the Geneva mechanism agent

Appendix C

Note on KQML

C.1 KQML based communication support

In order for an agent's actions to be meaningful, they need to be expressed in a language that could be interpreted by other agents. The development of Knowledge Query Manipulation Language (KQML) by Knowledge Sharing Effort (KSE) provides a means for formal inter-agent communication [Finin, *et al.*, (1995)]. Finin, *et al.*, pointed out three distinct components in the communication protocol. These are the content, message and transfer layers as shown in Figure C.1.

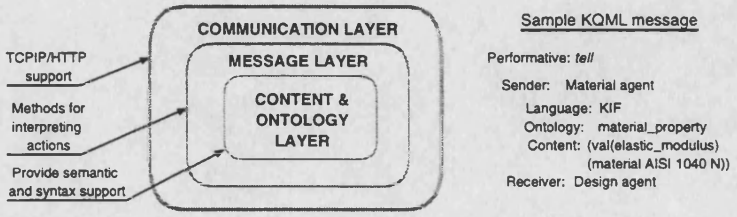


Figure C.1: KQML-based communication mechanism

The content layer represents the message in a specified language. For example, Labrou *et al.*, (1999) show that the content of a KQML message, including the

ontology to be shared, could range from simple ASCII strings to a more formal syntax using KIF, STEP or any other agreed format. Figure C.1 shows a sample KQML message with content expressed in KIF format. The message layer determines the kind of interactions the agent may choose to perform. This is governed by a set of *performatives* (e.g., tell, ask, agree) which is a term used to refer to the agent's action. The communication or transfer layer describes parameters such as the network addresses of the sender and recipient that are identified as unique names as shown in Figure C.1. This layer is supported by transport mechanisms such as TCP/IP or HTTP.

Major research is underway in adjacent areas such as a language for maintaining and building ontologies using Ontolingua [Gensereth and Ketchpel, (1994)]. JATLite developed at Stanford University and JAFMAS developed at University of Cincinatti, are some of the available templates for agent development and use KQML for support. KQML has also captured the attention of many companies such as British Telecom and Siemens [Labrou, *et al.*, (1999)].